Aslan Askarov
René Rydhof Hansen
Willard Rafnsson (Eds.)

LNCS 11875

Secure IT Systems

24th Nordic Conference, NordSec 2019
Aalborg, Denmark, November 18–20, 2019
Proceedings

Springer

Aslan Askarov · René Rydhof Hansen ·
Willard Rafnsson (Eds.)

Secure IT Systems

24th Nordic Conference, NordSec 2019
Aalborg, Denmark, November 18–20, 2019
Proceedings

Springer

Editors
Aslan Askarov
Aarhus University
Aarhus, Denmark

René Rydhof Hansen
Aalborg University
Aalborg, Denmark

Willard Rafnsson
IT University of Copenhagen
Copenhagen, Denmark

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-35054-3 ISBN 978-3-030-35055-0 (eBook)
https://doi.org/10.1007/978-3-030-35055-0

LNCS Sublibrary: SL4 – Security and Cryptology

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at the 24th Nordic Conference on Secure IT Systems (NordSec 2019). The conference was held during November 18–20, 2019, in Aalborg, Denmark.

The NordSec conference series started in 1996 with the aim of bringing together researchers and practitioners in computer security in the Nordic countries, thereby establishing a forum for discussion and cooperation between universities, industry, and computer societies. The NordSec conference series addresses a broad range of topics within IT security and privacy and over the years it has developed into an international conference that takes place in the Nordic countries. NordSec is currently a key meeting venue for Nordic university teachers and students with research interests in information security and privacy.

The 17 papers accepted for presentation, and included in the present volume, were carefully chosen from 32 submissions, all of which were double-blind reviewed by the Program Committee (PC). Additionally, a poster session was organized to encourage further discussion, brainstorming, and networking on interesting topics of IT security.

Finally, we would like to sincerely thank everyone involved in making this year's conference a success, including, but not limited to: the authors who submitted their papers, the presenters who contributed to the NordSec program, the PC members and additional reviewers for their thorough and very helpful reviews, and finally a thanks to EasyChair for their platform and support.

October 2019

Aslan Askarov
René Rydhof Hansen
Willard Rafnsson

Organization

Program Committee

Juha Röning	University of Oulu, Finland
Berry Schoenmakers	Eindhoven University of Technology, The Netherlands
Einar Snekkenes	Norwegian University of Science and Technology, Norway
Shukun Tokas	University of Oslo, Norway
Alix Trieu	Aarhus University, Denmark
Alexandre Vernotte	Femto-ST Institute, France
Øyvind Ytrehus	University of Bergen, Norway
Fabio Massimo Zennaro	University of Oslo, Norway
Bingsheng Zhang	Lancaster University, UK

Additional Reviewers

Chevrot, Antoine
Dexe, Jacob
Ul Hassan, Sohaib
Vajaranta, Markku

Contents

Privacy

Privacy Impact Assessment: Comparing Methodologies with a Focus on Practicality

Tamas Bisztray(✉) and Nils Gruschka

Department of Informatics, University of Oslo, Oslo, Norway
{tamasbi,nilsgrus}@ifi.uio.no

Abstract. Privacy and data protection have become more and more important in the recent years since an increasing number of enterprises and startups are harvesting personal data as a part of their business model. One central requirement of the GDPR is the implementation of a data protection impact assessment for privacy critical systems. However, the law does not dictate a special assessment methods.

In this paper we compare different data protection impact assessment methods. We have developed a comparison and evaluation methodology and applied this to three of the most widespread assessment frameworks. The result of this comparison shows the weaknesses and strength, but also clearly indicates that none of the tested methods fulfills all desired properties. Thus, the development of a new or improved data protection impact assessment framework is an important open issue for future work.

Keywords: Data protection · Privacy Impact Assessment · GDPR · DPIA

1 Introduction

Data is the new currency of the 21th century and there is an increasing number of businesses collecting and storing our personal information and making monetary benefits from it. The key to success for these businesses is to harness value from the collected data. To do this they need not just to store but to process what has been collected. Monetary benefits and business goals are easily pushing the protection of people's personal privacy down in the to-do list. The General Data Protection Regulation (GDPR) [8] was meant to give back the control to people over their private data.

Companies failing to fulfill requirements imposed by the GDPR can face serious fines laid out by Article 83 which in the worst case can be up to €20 million, or 4% of the firm's worldwide annual revenue from the preceding financial year, whichever amount is higher.

If a company wants to avoid such high fines they have to adjust their operations to become compliant with the GDPR. However, many worry that these regulations are imposing an unnecessary burden on tech companies. This opinion was also voiced by Alibaba's founder Jack Ma who said in an interview: "Europe will stifle innovation with too much tech regulation" [18]. Indeed for a startup such a fine would be devastating. Allocating too much resources to become fully compliant could be similarly harmful. Therefore, becoming GDPR compliant from the beginning without too much hassle is very important. But more than a year after the GDPR came into force there still exists no standard framework in the EU and companies are either doing an assessment on their own or they have to find out which DPIA method is the one that would suit their project the best.

Article 6(1) contains six requirements one of which is necessary to fulfill in order to lawfully process PII (personally identifying information). One possibility is to obtain consent from the data subject. Article 7 further states that consent shall be requested in a way that is: *"clearly distinguishable from the other matters, in an intelligible and easily accessible form, using clear and plain language".*

But what is intelligible, easy to understand plain language or what is appropriate in length can be matter of debate. Consent once acquired is very easy to demonstrate and allows the controller[1] to specify all use cases to rule out the possibility of unwanted legal issues. Whereas other requirements could be more difficult to prove.

Acquiring compliance therefore can look like one just have to ask something in a sophisticated way and if the user clicks accept the processing is lawful. Google's case serves as a counterexample where the company received a € 50 million fine, inflicted by French data protection authorities (CNIL) [9]. According to CNIL the way Google obtained consent violated the transparency of obligations and was lacking legal bases. This case sets a good example but unfortunately, in practice there are still many instances where the users are presented sophisticated documents where copyright claims and other legal matters are mixed with asking consent without a clear description of the processing purposes.

There are several rules the data controller has to follow. Following our example with consent Article 7(3) says "*The data subject shall have the right to withdraw his or her consent at any time*". If the data subject withdraws consent for storing his personal information the data controller shall delete it. Meanwhile if the data was copied, shared with several processors, has been processed after which new PII was created, all of this has to be deleted, too. This requires that the data controller should constantly keep track of the data.

Obtaining and maintaining compliance with the GDPR requires attention and a good overview of operations related to PII. Improper management of PII can lead to the violation of the GDPR. This is specially challenging for companies with complex systems designed prior to the GDPR. Tackling this problem requires a method guiding organisations to identify and document activities related to personal data and assess risks related to its processing, while

[1] Natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data.

providing steps to achieve compliance with the GDPR and the ability to demonstrate it. For this purpose Article 35 introduces DPIA (data protection impact assessment). Its a method that provides guided steps to identify and analyse how the rights and freedoms of individuals might be affected by certain actions or activities related to data processing, and to assist in avoiding/correcting these issues.

This paper will analyse and compare different DPIA methods. The goal of this work is twofold. Firstly, we are aiming to identify shortcomings of current DPIA methods. We will do this by proposing a metric that helps to identify advantages and disadvantages of existing methods. The second goal is to provide help for those who are planning to conduct a DPIA, but can't decide which framework would be the best for their application as of today. To help with this question we are briefly going through some frameworks highlighting their strengths and weaknesses.

Section 2 gives a general introduction to DPIA and an overview of the examined DPIA frameworks. Section 3 provides an overview of related work in comparing DPIA methods. In Sect. 4 we present our metric for comparing DPIA methods and perform the comparison itself. Section 5 contains the summary and conclusions.

2 Data Protection Impact Assessment

2.1 Legal Background

Data protection impact assessment is a new requirement under the GDPR as part of the *data protection by design and by default* principle (introduced in Article 25). According to Article 35:

> *If the processing is likely to result in a high risk to the rights and freedoms of natural persons, the controller shall, prior to the processing, carry out an assessment of the impact of the envisaged processing operations.*

Unfortunately it does not specify which type of processing requires a DPIA. The reader might be confused right from the start after finding these methodologies under the name of Privacy Impact Assessment (PIA). In practice they are the same concept. Originally PIA was only aiming to assess the privacy risks of a processing, whereas the GDPR requires DPIA to go beyond and determine sufficient security measures and safeguards to address these risks. As pointed out by Roger Clarke [12] in his *Comprehensive Interpretation of Privacy*, data privacy is just one of the four aspects of privacy where the other three are: privacy of the person, privacy of personal behaviour and privacy of personal communications. In this paper we prefer the use of DPIA over PIA but they are treated as synonyms.

In the *Guidelines on Data Protection Impact Assessment* published by the Article 29 working party nine criteria for processing likely to result in a high risk scenario (and therefore requiring a DPIA) are defined [1]. Similarly there is a list of cases in which no DPIA is required. The general guideline is that if you are not sure where a certain process belongs a DPIA has to be performed.

Furthermore, Article 35(11) requires that a new DPIA has to be carried out when there is a change related to the risk of the process. This means that processes must be tracked over time in order to detect these kind of changes. This also applies to processes which are at the moment labeled low risk since the risk might change to high.

The GDPR does not reference a concrete DPIA method, but Article 35(7) at least defines the minimal content of a DPIA:

- *a systematic description of the envisaged processing operations and the purposes of the processing, including, where applicable, the legitimate interest pursued by the controller;*
- *an assessment of the necessity and proportionality of the processing operations in relation to the purposes;*
- *an assessment of the risks to the rights and freedoms of data subjects referred to in paragraph 1;*
- *the measures envisaged to address the risks, including safeguards, security measures and mechanisms to ensure the protection of personal data and to demonstrate compliance with this Regulation taking into account the rights and legitimate interests of data subjects and other persons concerned.*

Therefore, no DPIA methodology can miss any of these points.

The Article 29 working party recommends EU generic DPIA frameworks from 4 countries (DE, ES, FR, UK) [1] and two EU sector-specific frameworks "Privacy and Data Protection Impact Assessment Framework for RFID Applications" [6] and "Data Protection Impact Assessment Template for Smart Grid and Smart Metering systems" [7]. The ISO/IEC 29134 as international standard is referenced.

For applying our DPIA comparison approach, we will select the DPIA methods from this list. Additionally we will use LINDDUN as the most well-known DPIA framework. We used the following criteria when selecting the methods to compare:

- The method has recently been updated.
- An English version is available.
- The method offers a good selection of supporting material.
- From each of these origin at least one framework is selected: policy-driven, academic, international.

Based on these criteria, the following methods have been selected for detailed description and comparison: LINDDUN, CNIL, ISO/IEC 29134:2017. In the following sections these methods are presented in more detail. Throughout the analysis remarks will be made in the form of numbered notes to underline positive or negative aspects of each method. These will be referenced in the evaluation.

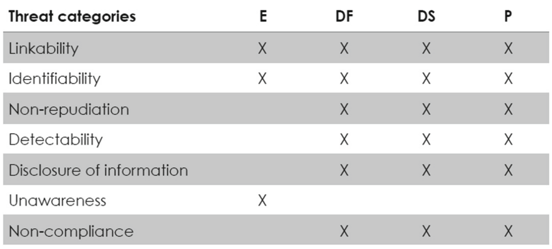
2.2 LINDDUN

LINDDUN is a privacy threat analysis framework, developed by researchers from the DistriNet Research Group at KU Leuven, Belgium [17]. LINDDUN, is an acronym for Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, Non-compliance and consists of 6 main steps.

1. Define Data Flow Diagram (DFD). It uses a data flow diagram to provide a high level graphical description of the whole architecture (based on SDL threat modeling). In a system model there are 4 different building blocks: entity, process, data flow, data store.

Note 1: This representation makes it possible to differentiate between threat analysis (for incoming and outgoing information) and privacy analysis (for internal data flow and processes) which is very helpful. It is also useful to map existing architecture, for example Data store = Data base.

2. Mapping Privacy Threats to DFD. This step helps to identify threats connected to each element in the DFD. Figure 1 shows the LINDDUN mapping table, where each row is a LINDDUN threat category and the columns contain all DFD elements. If a threat is relevant to a DFD type it is marked with X.



Threat categories	E	DF	DS	P
Linkability	X	X	X	X
Identifiability	X	X	X	X
Non-repudiation		X	X	X
Detectability		X	X	X
Disclosure of information		X	X	X
Unawareness	X			
Non-compliance		X	X	X

Fig. 1. Mapping privacy threats to DFD element types (E-entity, DF-data flow, DS-data store, P-process) (Source: [17])

3. Identify Threat Scenarios. Each X in the table has to be examined to determine whether they pose a threat to the system. Lindunn provides a set of privacy threat threes to each X. (With the exception of Disclosure of Information, where LINDDUN points to STRIDE for further analysis). If the threat is relevant misuse cases has to be documented from the misactors point of view. Otherwise the assumptions on why something is not relevant has to be documented.

4. Prioritise Threats. There can be an overwhelming number of threats and due to budget and time limitations first (or only) the most important are considered. Risk assessment is not part of LINDDUN and it offers a number of methods to perform this step: OWASP's Risk Rating Methodology [19], Microsoft's DREAD [20], NIST's Special Publication 800-30 [21], or SEI's OCTAVE [22].

Note 2: It would be useful to have an improved version/combination of these tools tailored for LINDDUN so that the security analyst conducting this step doesn't have to figure out which method would suit his application the best.

5. Elicit Mitigation Strategies. Every threat tree is automatically linked to a mitigation strategy (which later is linked to solution steps). In a case study where LINDDUN was used for DPIA with a Identity Wallet Platform [14] the authors considered this impractical and used the ISO/IEC 27005 *Information Security Risk Management*, which identifies four mitigation strategies: risk reduction, retention, avoidance, and transfer.

6. Select Corresponding Threats. For every mitigation strategy a list of related papers is provided on appropriate privacy enhancing technologies. This is organised as a table and can be found in the supporting materials. In [14] the authors noted that they faced "lack of expertise, low technology readiness level, and other uncertainties regarding the integration of Privacy-Enhancing Technologies (PETs)". They also identified further PETs not present in the table. They also described the need of finding balance between project goals and privacy goals as they had trouble addressing all privacy threats.

2.3 CNIL

The CNIL PIA framework was created by the French data protection authority. They published their DPIA method [3] in November 2018 incorporating the GDPR and the Article 29 Working Party's opinion [1]. It has a very well rounded list of supporting materials: Methodology, Knowledge bases, Templates, Application to IOT devices examples of data processing operations likely to result in a high risk [2] and a software tool that helps to go through the steps [10]. It helps to demonstrate compliance and provides guiding steps to achieve it. Compliance is defined as a combination of the following two pillars: Fundamental Rights and Principles (non-negotiable), Management of data subjects/privacy risks (technical controls). It consists of 4 steps: describe context of processing, guarantee compliance with fundamental principles, assess privacy risks and treat them, and document validation.

1. Study of Context. The main steps here are:

- outline processing
- identify data controller and any processor
- check applicable references: approved codes of conduct (Article 40), certifications regarding data protection (Article 42)
- define personal data concerned / recipients, storage duration
- describe processes and personal data supporting assets.

Note 3: As a first step it doesn't give a good picture of the whole architecture, data movement is not visible and doesn't allow to later differentiate threat analysis and privacy analysis. The online tool doesn't support grouping or any structuring of this information. On the upside it references the GDPR and the relatable sections to help with compliance and clearly defines what is expected at each entry and what the conductor of the DPIA should pay attention to.

Note 4: Article 35(1) requires "a systematic description of the envisaged processing operations". This could be a matter of debate but in contrast to LINDDUN CNIL doesn't fully comply with this point. It does list and describe the processing operations but the process itself doesn't lay out a systematic way on how this should be conducted.

Note 5: Very important to remark: CNIL in its collection of supporting materials among which is a document called "Templates", does provide tables to collect and categorise information. It is not as good as a visual representation but good for record-keeping. However, this paper focuses on the process and the practicality of the DPIA method. If during the steps of the DPIA it is not explicitly mentioned or referenced if something additional is needed for that very step, or the concept is not present in the main document which describes the process, it will not be counted as part of the process. The DPIA should be intuitive with sufficient guidance. Due to the plethora of templates available it is not easy to get a hold of what is needed for a certain step. The main document itself never references to any of the templates and their lack of integration into the process is a serious drawback.

2. Study of Fundamental Principles. The aim is to ensure compliance with privacy protection principles. It consists of two steps: "Assessment of the controls guaranteeing the proportionality and necessity of the processing to enable the persons concerned to exercise their rights" and "Assessment of controls protecting data subjects' rights".

Note 6: This step is a direct translation of the second bare minimum principal from Article 35 (7b) and it is a simple compliance check.

3. Study of Risk. *Note 7*: So far the software tool and the written material were in sync, the tool used the same points described in the text but from this point on it forks into two different processes with different questions.

The document first gives a general introduction to risk assessment with a brief overview on how to calculate risk level. The supporting material *Knowledge Bases* provides a very detailed tutorial on how to determine severity and likelihood with a lot of real life threat scenarios. Study of risks contains two sections. The first one is *Assessment of existing or planned controls* on controls of data being processed: encryption, anonymization etc., general security controls, and organisational controls.

Note 8: The order of the steps so far are incorrect in the documentation. Potential threats were not yet identified neither controls mitigating those threats. Encryption is used if for example confidentiality needs to be protected, but there are many forms of encryption and first a threat needs to be recognised to use the proper countermeasures.

The second section is on *Risk assessment.* It requires for each impending event:

- determine potential impact on the data subject privacy
- estimate severity of impact
- identify threats to personal data supporting assets, that leads to this feared event (threat scenario) and the risk sources (threat actors)
- estimate likelihood
- determine whether the risks identified in this way can be considered acceptable in view of the existing or planned controls.

Note 9: These steps are out of order. Likelihood and severity has to be calculated before impact.

Note 10: Considering if a risk can be acceptable doesn't qualify as prioritising threats.

The software tool interestingly follows a different steps: Planned or existing measures, illegitimate access to data, unwanted modification of data, data disappearance, and risks overview. It eerily resembles the CIA triad (confidentiality, integrity, availability) with "Planned or existing measures" and "Risk overview" added. It also doesn't categorise the risk properly, neither differentiates between non-negotiable and technical controls.

4. Validation of the DPIA. In a timely fashion this section correspond to Article 35 (7d): "the measures envisaged to address the risks, including safeguards, security measures and mechanisms to ensure the protection of personal data and to demonstrate compliance...". It consists of 3 steps: prepare material, formal validation, repeat when necessary (no further comment on how often).

2.4 ISO/IEC 29134:2017

The ISO/IEC standard number 29134:2017 also has a nice selection of supporting material, mainly other ISO standards like a risk-based management system: ISO/IEC 27001, or overview and vocabulary: ISO/IEC 29100:2011 etc. The document itself starts with a long discussion on principles and guidelines related to conducting the DPIA such as: preparing grounds, benefits of DPIA, objectives of reporting, accountability to conduct, scale, determine if DPIA is necessary, preparations, set up a team, prepare a plan, describe what is being assessed, and stakeholder engagement.

The actual steps of the DPIA only starts at Sect. 6.4 and it consists of 5 main steps:

1. Identify information flows of personally identifying information (PII)
2. Analyse the implications of the use cases
3. Determine the relevant privacy safeguarding requirements

4. Assess privacy risks
5. Prepare for treating privacy risks.

The general structure for these steps consists of the following list the conductor has to fill out:
Objective, *Input*, *Expected output*, *Actions*, *Implementation guidelines*.

The guiding document provides a detailed description of what is expected at each point of this list that is tailored to the main steps. After the list it provides a detailed guide with comments and recommendations to further assist the conductor of the DPIA, by for example listing organisational and non-compliance threats and other tips related to that part of the assessment. *Note 14:* This approach makes the whole process a bit monotone almost like filling out a questionnaire but at least it discusses the important topics.

The next sections are the DPIA follow up and the DPIA report. These are more detailed than the DPIA itself. For example the Risk assessment section contains: Threats and their likelihood, Consequences and their level of impact, Risk evaluation, Compliance analysis. But it always references back to previous points so it doesn't mean it was missing from the process. Plus it is easy to put together and provides a very detailed report.

3 Related Work

Although there is an overabundance in available DPIA methods there hasn't been a lot of work on evaluating and comparing them. Further, from the existing literature only a small proportion was published after the GDPR came into force. There are two types of papers in this topic. Evaluating DPIA methods and comparing/measuring the effectiveness of DPIA reports. These are two different fields but in the pursuit of evaluation a lot can be learned from the study of reports as well.

As mentioned before, the GDPR unfortunately does not provide an actual framework to follow and it also doesn't recommend one. This is a shortcoming recognised by Wright et al. already in 2013 [16]. They urged that the European commission and EU member states should draw from the experience of other countries and develop their own DPIA policy, methodology and framework. They also pointed out that a DPIA should be more than a compliance check, as it should be a process. It has to be reviewed and updated throughout the whole life cycle of the project as also stated in Article 35(11). They compared DPIA methods from six countries drawing inspiration from the PIAF project [11] cofounded by the European Commission which reviewed DPIA methods from other countries.

The PIAF delivereable 1 compared the effectiveness of DPIA guides based on a checklist of 18 questions [4, table 10.1] and comparing DPIA reports using checklist of 10 questions [4, table 10.2]. The thirds deliverable [5] outlines what a DPIA process should conatin. These are: Project description, Stakeholder consultation, Risk management, Legal Compliance check, Report, Implementation, Review.

The RFID framework [6] even though it is sector-specific, recognises eight important steps a DPIA should address. These are: characterization of the application, initial analysis, definition of privacy targets, evaluation of degree of protection demand for each privacy target, identification of threats, identification and recommendation of controls, consolidated view of controls, assessment and documentation of residual risks. The frameworks our analysis will focus on are those designed for general use-cases.

So far for comparing DPIA methods the most commonly used technique was to go through a checklist to see if it fulfills certain requirements. However, this can be only done if the evaluation criteria is quantitative in nature and only checks the existence or non existence of a certain aspect. For example in the comparison of Wright et al. one checkpoint is: "Provides a suggested structure for the PIA report". This is a binary question. Whereas other qualitative matters shouldn't be written off with a check-mark. For example points like *DPIA is a process* or *DPIA is more than a compliance check*. It doesn't matter if a method says or claims these points, the real question is how well they fulfill these requirements. It is also pointless to include such questions in a DPIA report analysis as it is not the job of the project owner to invent a working DPIA method that has a nice workflow, rather it's the task of those developing DPIA methods and tools to fulfill these requirements. A report is a statement on what has been performed. These problems are commonly present in previous works by either treating important question as a check-mark and not uncovering shortcomings of the DPIA method, or including questions related to the quality of the method in the report analysis.

An improvement to the check-mark approach was PEGS (PIA Evaluation and Grading System) proposed by Whadwa et al. [15]. Even though this method was developed to evaluate the actual DPIA process post facto (the DPIA report) not the DPIA itself, the authors note that it can be helpful also in guiding a DPIA process. Their evaluation criteria is first presented as a checklist where they provide an extra column where in case a requirements was not fulfilled *scope of improvement* can be specified. Then a weighting is applied in three categories 1, 2 or 3 to each criteria in line with their relative importance, where 1 is the least important and 3 is the most important. Their choice of weights was highly based on the PIAF project. Criteria with weight 1: clarification of early initiation, identification of who conducted the DPIA and publication; weight 2: project description, purpose and relevant contextual information, information flow mapping, legislative compliance checks and identification of stakeholder consultation; weight 3: identification of privacy risks and impacts, identification of solutions/options for risk avoidance and mitigation, and recommendations handling after the PIA.

In a more recent analysis Vemou et al. [13] reviewed 9 different DPIA methods regardless of country of origin with the only criteria that it should not be sector specific, by using 17 questions derived from existing literature as check-marks to draw attention the to lack of completeness of these methods.

4 Comparison of DPIA Methods

4.1 Comparison Metric

To successfully evaluate a DPIA method we should differentiate between three types of criteria. Questions that relate to steps someone should consider prior starting the DPIA are preliminary questions, they are important to consider but they don't have to be integrated into the process. Similarly there are a series of questions only relevant after a DPIA is completed. Like "is publication of the DPIA report provisioned". These are in the territory of DPIA report analysis (which is the scope of our future research) and again is not something that needs to be part of the DPIA process itself. The questions the we focus on are the ones directly related to the steps of DPIA. Questions should also point out the shortcomings identified in the *Notes*. Some questions such as "is it a process" or "is structured guidance to assist in risk assessment provided?" cannot be answered with a single check-mark as they are rather qualitative questions. However, it would be very difficult to build a metric with open questions. In previous works questions related to the DPIA process were simply listed. Here the questions will be structured based on which part of the DPIA process should contain it. It's important to note that Article 29 working party's document is the only official recommendation on how the process should look like. Therefore, we consider that as a starting point[2] .

In total there will be 28 questions. These are categorised based on which part of the DPIA they belong to. The evaluation happens the following way. Each questions will receive grades in two categories: *score* (S) and *process* (P). The Score (S) meant to determine if the question is covered by the method in general. For the score a question can get 0 points if it is not in the method, 1 point if it is partially included, and 2 if its completely addressed. The Process (P) meant to evaluate if a question is well integrated in the DPIA process, for example is discussed in the right part of the DPIA, which can also mean it's part of the method but not placed correctly or logically from the perspective of the whole process. Is it properly discussed with the necessary supporting materials included. For the *process* it can get +1 if the step is integrated into the process or −1 if it is not integrated). For a zero Score the Process is automatically zero too. This approach can penalize if a framework while mentioning a certain criterions or questions doesn't integrate its steps into a process and it's closer to a compliance check. For a simple check-mark evaluation this is not possible.

[2] Their "Criteria for acceptable DPIA" checklist, however, is not a blueprint for a good process. The points of Article 35 (7) of the GDPR was also only meant to be a list. Unfortunately, the steps of CNIL is almost a point to point copy of these two.

In this analysis we are not going to focus on a complete compliance checklist but many related questions are included in the process. Checking all the applicable points of the GDPR is not the scope of this paper, as it is more important during the report evaluation.

4.2 Evaluation Questions

This section contains the grading questions split up between six tables for the six steps we identified as crucial parts of the DPIA process. The cells apart from the received grades in some cases also contain a reference to the *"Notes"*. For example *Note 1* is denoted as $^{(1)}$. Grades for (S) and (P) are also separately shown *(Breakdown)* before summarised in the *Total* score where the maximum achievable grade is also shown. Most questions are aiming to evaluate the content of the methods, while others are specifically trying to uncover if the order of steps and questions in the method are designed properly.

Step 1: Description of envisaged processing	ISO		CNIL		LIN.	
	S	P	S	P	S	P
Structured description and mapping of information flows, contextual information and envisaged processing (structured: either graph or table)	2	+1	2	−1 (3) (4) (5)	2	+1 (1)
Establish easy to follow connections between system elements (data, process, supporting assets etc.)	2	+1	1	−1 (5)	2	+1 (1)
Allow the differentiation of internal and external data movement	2	+1	0	0	2	+1 (1)
Stakeholder identification	2	+1	2	+1	1	+1
Breakdown	8	+4	5	−1	7	+4
Total score (out of 12)	12		4		11	

Both LINDDUN and ISO are using visual representation of the information flows and they are described during the process. LINDDUN is more intuitive but the instructions provided in ISO are more detailed. Unfortunately, the CNIL method really falls behind at this point, which is a very serious issue. This step is the foundation stone for the whole DPIA and missing points here is a serious problem.

Step 2: Assess necessity and proportionality of processing	ISO		CNIL		LIN.	
	S	P	S	P	S	P
How information is to be collected, used, stored, secured and distributed and to whom and how long the data is to be retained	2	+1	2	+1	0	0
Compliance with Article 29 Working party's corresponding list (Annex 2/necessity and proportionality)	2	+1	2	+1	0	0
Analyse all previously identified system elements in a structured manner	2	−1	0 (6)	0 (6)	2	+1
Breakdown	6	+1	4	+2	2	+1
Total score (out of 9)	7		6		3	

LINDDUN is clearly missing assessment steps from a legal perspective and mainly focuses on threat analysis. CNIL does a perfect job from a compliance perspective but it doesn't connect the dots. For the next step we deviate from the suggestion of Article 29 Working Party which would be: "measures envisaged". To determine how the information has to be stored and secured it is important to know the context, from who it has to be protected. Here we prefer the approach of LINDDUN where an early threat analysis is initiated.

Step 3: Identify threats/risks	ISO		CNIL		LIN.	
	S	P	S	P	S	P
Organisational and technical that are endangering the rights of data subjects	2	+1	2	+1	0	0
Origin of risks are specified (threat actor-attack surface)	2	−1	2	−1	2	+1
Threats should be directly linked to elements from step 1	1	−1	1	−1	2	+1
Identification of threats coming from GDPR non-compliance is integrated into the process	2	+1	2	+1	0	0
Threats are identified before Risk Assessment	2	+1	0 (8)	0	2	+1
Is there a differentiation between threat analysis and privacy analysis	2	+1	2	+1	1	−1
Addresses all types of privacy risks (informational, bodily, territorial, locational, communications)	1	+1	1	−1	1	+1
Breakdown	12	+3	10	0	8	+3
Total score (out of 21)	15		10		11	

The ISO standard proves to be the best in this case as well. The drawback of LINDDUN is again the fact that legal compliance is not integrated in the process, which the authors clearly state in the beginning as a result lot of aspects are missing (although some are accidentally tackled). LINDDUN only considers a limited number of threats. CNIL is very strong contentwise but there is no logical structure in its steps and it is a simple compliance check.

Step 4: Risk Assessment	ISO		CNIL		LIN.	
	S	P	S	P	S	P
Structured guidance to assist in risk assessment is provided	2	+1	2	+1	0	0
Fundamental Rights and Principles (non-negotiable) and Management of data subjects/privacy risks (technical controls) are differentiated	1	+1	2	+1	0	0
Risk calculation is included with sufficient supporting material	2	+1	2	−1 (9)	0	0
Risks are prioritised	2	+1	1 (10)	+1	0	0
Lower risks that are not immediately addressed are well documented	2	+1	1	+1	2	+1
Risk reduction, retention, avoidance, and transfer are all listed as mitigation strategies and sufficiently discussed in supporting material	2	+1	1	+1	0	0
Owner of residual risks specified	2	+1	2	+1	0	0
Breakdown	13	+7	11	+5	2	+1
Total score (out of 21)	20		16		3	

LINDDUN doens't include a risk assessment, only recommends some. It gets some points because in the previous step all threats were already documented. ISO also points out to other ISO/EIC standards and guides, but it does include a structured guide on its own and the recommendations are well referenced and compatible. Whereas LINDDUN leaves the privacy analyst alone to figure out which method would be the best for his use case. If CNILs software tool would be also evaluated for this step its score would be closer or below LINDDUNs.

Step 5: Measures envisaged	ISO		CNIL		LIN.	
	S	P	S	P	S	P
Technical controls and PETs are only discusses after threats and related risks have been evaluated	2	+1	1	−1	2	+1
An extensive list of organisational measures are provided	1	+1	2	−1	0	0
An extensive and updated list of technical measures (PETs) are available	1	−1	2	−1	2	+1
Literature/supporting material for suggested PETs are included	0	0	2	−1	2	+1
Breakdown	4	+1	7	−4	6	+3
Total score (out of 12)	5		3		9	

CNIL again is very vague in the main document but, the fact that in it's "knowledge bases" supporting material provides a wide selection of technical measures none of which is referenced in the process. The list provided by LINDDUN can not be considered complete (neither CNILs or ISO), but it's a step in the right direction.

Step 6: Documentation/Validation	ISO		CNIL		LIN.	
	S	P	S	P	S	P
Outline of the report was generated during the process	2	+1	1	−1	1	−1
Result is evaluated	2	+1	2	+1	0	0
Action plan for continuation	2	+1	2	+1	1	−1
Breakdown	6	+3	5	+1	2	−2
Total score (out of 9)	9		6		0	

Here ISO outruns the other methods in terms of DIPA report preparation. The steps are already outlined in the main document and every step is referenced back to a step from the process.

Evaluation	ISO		CNIL		LIN.	
	S	P	S	P	S	P
Final Breakdown	49	19	42	3	27	10
Final Score (out of 84)	68		45		37	

The overall result shows that, CNIL lost a lot of points for coming off as a compliance check and not trying to be a better process, and due to the lack of references. The ISO method proved to be the best but it could also use a bit if improvement as the order of its steps and the content are good, it feels like a questionnaire and not a genuine process. LINDDUN needs to develop a step for risk assessment and documentation, while steps and references for GDPR compliance must be incorporated throughout the process.

5 Summary and Outlook

In this paper we performed a comparison of widespread data protection impact assessment methods. By approaching the evaluation and grading from the perspective of a process rather then a compliance check, it became obvious that in many cases very important points of these methodologies are not properly worked out. The ISO standard proved to provide the best framework both contentwise and as a process, although there are still many shortcomings waiting for improvement. The latter is even more true in case of CNIL and LINDDUN. These are among the state of the art DPIA methods with the purpose of: helping companies implementing the Privacy by Design paradigm, support developing GDPR compliance (not least to avoid fines such as Google got), but mostly to assist in the protection of the rights and freedom of natural persons. CNIL has a very good selection of supporting material and in terms of achieving GDPR compliance, it is the best method to go for. However, as a process it really doesn't perform well. LINDDUN has a very good start but it completely misses Risk Assessment and it's 5th step (Eliciting mitigation strategies) is not very intuitive and it's not strong on documentation/validation.

Following Wright et al. [16] we also highlight the importance of one or more officially approved EU-specific DPIA frameworks with sufficient and regularly updated supporting material. In future work we will apply these frameworks to various projects to address the question of GDPR compliance more deeply and analyse the DPIA reports, with the intention of proposing improvements to these methods.

References

1. Article 29 Working Party: Guidelines on Data Protection Impact Assessment (DPIA) (2017). https://ec.europa.eu/newsroom/article29/item-detail.cfm?item_id=611236
2. Commission Nationale de l'Informatique et des Libertés: Analyse d'impact relative à la protection des données : publication d'une liste des traitements pour lesquels une analyse est requise (2018). https://www.cnil.fr/fr/analyse-dimpact-relative-la-protection-des-donnees-publication-dune-liste-des-traitements-pour
3. Commission Nationale de l'Informatique et des Libertés: Privacy Impact assessment (pia) — CNIL (2019). https://www.cnil.fr/en/privacy-impact-assessment-pia
4. Wright, D., Wadhwa, K., De Hert, P., Kloza, D.: A Privacy Impact Assessment Framework for data protection and privacy rights (2011), https://piafproject.files.wordpress.com/2018/03/piaf_d1_21_sept2011revlogo.pdf
5. De Hert, P., Kloza, D., Wright, D.: Recommendations for a privacy impact assessment framework for the European Union (2012). https://piafproject.files.wordpress.com/2018/03/piaf_d3_final.pdf

6. European Commission: Privacy and Data Protection Impact Assessment Framework for RFID Applications (2011). https://ec.europa.eu/digital-single-market/en/news/privacy-and-data-protection-impact-assessment-framework-rfid-applications
7. European Commission: Smart Grids Task Force (2014). https://ec.europa.eu/energy/en/topics/markets-and-consumers/smart-grids-and-meters/smart-grids-task-force
8. European Parliament & Council: Regulation (EU) 2016/679 - Regulation on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union L119 (4.5.2016), 1–88 (2016)
9. Quathem, K.V., Tielemans, J., de Meneses, A.O., Shepherd, N.: Google fined EUR 50 million in France for GDPR violation (Jan 2019). https://www.insideprivacy.com/eu-data-protection/google-fined-e50-million-in-france-for-gdpr-violation/
10. de l'Informatique et des Libertés, C.N.: The open source PIA software helps to carry out data protection impact assesment (2019). https://www.cnil.fr/en/open-source-pia-software-helps-carry-out-data-protection-impact-assesment
11. PIAF: A Privacy Impact Assessment Framework for data protection and privacy rights (2011). https://piafproject.wordpress.com
12. Clarke, R.: Roger Clarke's 'Privacy Introduction and Definitions' (2016). http://www.rogerclarke.com/DV/Intro.html
13. Vemou, K., Karyda, M.: An Evaluation Framework for Privacy Impact Assessment Methods. In: MCIS 2018 Proceedings (2018)
14. Veseli, F., Olvera, J.S., Pulls, T., Rannenberg, K.: Engineering privacy by design: lessons from the design and implementation of an identity wallet platform. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing - SAC 2019, pp. 1475–1483. ACM Press, Limassol, Cyprus (2019). http://dl.acm.org/citation.cfm?doid=3297280.3297429
15. Wadhwa, K., Rodrigues, R.: Evaluating privacy impact assessments. Innovation: Eur. J. Social Sci. Res. **26**(1–2), 161–180 (2013). https://doi.org/10.1080/13511610:2013:761748
16. Wright, D., Finn, R., Rodrigues, R.: A comparative analysis of privacy impact assessment in six countries. J. Contemp. Eur. Res. **9**(1), 21 (2013)
17. Wuyts, K., Joosen, W.: LINDDUN privacy threat modeling: a tutorial. CW Reports (2015)
18. Soo, Z.: Alibaba's Jack Ma says he is 'worried' Europe will stifle innovation with too much tech regulation — South China Morning Post (May 2019). https://www.scmp.com/tech/big-tech/article/3010606/alibabas-jack-ma-says-he-worried-europe-will-stifle-innovation-too

Rotten Cellar: Security and Privacy of the Browser Cache Revisited

Florian Dehling(✉), Tobias Mengel, and Luigi Lo Iacono

TH Köln University of Applied Sciences, Cologne, Germany
{florian.dehling,tobias.mengel,luigi.lo_iacono}@th-koeln.de

Abstract. Web browsers use HTTP caches to reduce the amount of data to be transferred over the network and allow Web pages to load faster. Content such as scripts, images, and style sheets, which are static most of the time or shared across multiple websites, are stored and loaded locally when recurring requests ask for cached resources. This behaviour can be exploited if the cache is based on a naive implementation. This paper summarises possible attacks on the browser cache and shows through extensive experiments that even modern web browsers still do not provide enough safeguards to protect their users. Moreover, the available built-in as well as addable cache controls offer rather limited functionality in terms of protection and ease of use. Due to the volatile and inhomogeneous APIs for controlling the cache in modern browsers, the development of enhanced user-centric cache controls remains—until further notice—in the hands of browser manufacturers.

Keywords: Browser cache · Security · Privacy

1 Introduction

Large distributed systems such as the web require technologies that provide high scalability. A mechanism that enables to reduce the amount of data to be transferred between the client and the origin server drastically is the caching of resources. Besides web caching systems, such as proxy caches and content distribution networks (CDN), the browser-internal cache plays an important role in reducing the amount of data transmitted over the network. This is done by storing static content temporarily on the user's system and loading it as required. The consequence of reusing a file at a later time is the question of its validity and freshness. Even though the browser cache has not yet achieved public fame in terms of security and privacy risks, several attack strategies exist (e.g. [10,16,26]) that emphasise the urgency to look at this topic more thoroughly.

This paper contributes towards this claim by capturing the current usage of HTTP browser caches, summarising security and privacy issues, as well as pointing out the necessity for future research on this topic. For this purpose, experiments are performed to explore the influence of browser caches on quality of

A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 20–36, 2019.
https://doi.org/10.1007/978-3-030-35055-0_2

service metrics, as well as to analyse the content of the browser cache in relation to shared resources. Moreover, the browser cache's handling of TLS-protected resources is investigated by carrying out a repetition experiment. More specifically, it is examined how browser caches handle resources that are transferred with different TLS certificates for the same origin domain. Finally, approaches and tools are investigated that allow users to improve their security and privacy when using the browser cache.

The remainder of this paper is structured as follows. First, the foundations of HTTP caching are laid with a particular focus on the browser cache. In Sect. 3, possible risks for security and privacy of users are summarised. In order to record the current use of the browser cache, Sect. 4 describes several experiments that were conducted and discusses the obtained results. Amongst them are the achievable data transfer savings as well as the files that are reused from the browser cache likewise by a large number of websites. Another experiment investigates the relationship between the validity of TLS certificates and the behaviour of the browser cache. Possibilities for users to influence the behaviour of the browser cache are discussed in Sect. 5. Beside comparing the procedures to completely delete the browser cache of well-known web browsers, available browser add-ons for extended control of the cache are examined. Last but not least, Sect. 6 discusses the results and presents some suggestions that could contribute to improving a user's security and privacy when browsing the web while maintaining the benefits of browser caching.

2 HTTP Caching Background

Transmitting data in large distributed systems can be expensive, especially if the bandwidth is limited somehow. As modern websites consists of numerous files, technologies are used that reduce the amount of data to be transmitted. The strategy considered here stores static components of a website on the clients system, in order to be able to call them up if necessary without renewed transmission. While so-called caches are used in numerous types on the internet, this paper is limited to the HTTP browser cache.

If a website contains static content like e.g. images or scripts, web browsers can store those files locally inside their respective browser cache. Once the client requests the corresponding website again, previously stored files can be loaded from the cache instead of requesting and transmitting them from the origin server again. This technology is part of the *Hypertext Transfer Protocol*, specified by RFC 2616 [12] and revised in RFC 7234 [11]. As not every resource of a website is suitable to be cached and even the content of supposedly static files may changes over time, a web server can send instructions on how the browser cache should behave and handle cached resources. This is done via the `Cache-Control` header in the corresponding HTTP response. In the following, the basic directives are listed and described briefly.

`no-cache`. Forces caches to deliver the request to the source server to check the validity of each request before releasing a cached copy.

`no-store`. The browser is strictly forbidden to temporarily store the response from the server. This statement is usually used if the response contains sensitive data.

`private/public`. Using the instruction `private`, responses can only be stored in the browser cache as they may contain private information. `Public` allows caching in shared caches, even if the response requires HTTP authentication.

`max-age`. This statement specifies the maximum time period in seconds beginning with the time of the initial request in which a stored response can be reused. If for example a response is assigned with the Cache-Control header directive `max-age=86400`, the corresponding resource can be reused up to one day.

Once a resource has been stored in the browser cache, the question arises as to whether this file is up-to-date. As specified in RFC 7234, two procedures can be used for this kind of verification and which enable a browser to perform so-called *explicit* caching. The *freshness lifetime* rates the validity of a resource by features which were previously defined by the web server. The HTTP header `Expires` or the `Cache-Control` header directive `max-age` can be used to define a period of time within which a resource may be reused by caches. A *freshness validation* describes a process to check the validity of a resource by consulting the web server but without transferring the actual payload. Using conditional requests [13], indicators are transmitted that provide information about the version of the cached resource. This can be done using the `ETag` header, which contains an opaque string describing the version of the resource, or by using time information such as the `Date` or `Last-Modified` header. One of these three features can be used by the client to send a conditional request to the server. If the latest version of the requested resource is present at the client's cache, the server responds with a status code `304 Not Modified`. If the resource invalidated in the meantime, it will be retransmitted in the response body.

Missing validation-features could theoretically cause a client to store a file for an indefinite time. To prevent this, web browsers perform so-called *implicit* caching if the server does not provide any features for *freshness lifetime* or *freshness validation*. The procedure of implicit caching depends on the implementation of the web browser and has been investigated in [22].

3 Security and Privacy Implications of the Browser Cache

In addition to the advantages that browser caches offer to users, they can also be a gateway to cyberattacks. These can be divided into two types, (1) those affecting the user's privacy and (2) those placing malicious code on the victim's computer. Several possible attack vectors have been detected and discussed in the literature already and are summarised in the following.

3.1 Browser Cache Poisoning

Browser Cache Poisoning attacks consists of placing manipulated files to the affected browser cache [15,19,26]. The procedure is as follows. The victim first visits a website which is described by an HTML document that also includes instructions to load further documents like JavaScript files. The attacker replaces one of those additional resources with his poisoned version. This can contain malicious code that gets executed if the document is interpreted by the victim's web browser. To infiltrate his modified file, the attacker can make use of a Man-In-The-Middle attack (MITM). This enables him to compromise the communication between the victim and the web server which is requested to deliver the content for the appropriated website. Beside manipulating the actual content, the attacker also ensures that the Cache-Control header enables the browser cache to store the file as long as possible without performing an validation process. If the victim visits the same website again at a later point of time, the web browser will load the manipulated file from the browser cache and executes the malicious code. At this time, the attacker does not need to have access to the communication between the victim and the requested web server as the malicious file was stored at the victims browser cache at the time of the actual attack. In order to achieve greater impact, the attacker will try to compromise files that are used by multiple websites likewise and thus requested by the victims browser more frequently. Files that are distributed via CDNs are a promising solution for this purpose. The risk of such an attack increases when resources are transmitted via unsecured connections.

3.2 User Tracking

Beside Browser Cache Poisoning attacks, the browser cache offers vulnerabilities that affect the user's privacy. This can be done in several ways. One approach is to place user-based-identifiers in HTML documents. In this case, a server-defined ID is assigned to e.g. a hidden DOM-element. If the appropriated HTML document is cached by the victims web browser, it will be loaded containing the identifier, when the victim is visiting the website at a later time. Using a script, the embedded ID can be extracted and transmitted to the server. On this way, both page visits can be associated and the user is identified [14].

Analysing loading times, also called Cache Timing Attacks (CTAs), is another possibility to track users online activities with the help of the browser cache [8]. This approach makes usage of one of the core features of this technology, the drastic shortening of loading times. In order to explore which websites already have been visited by the victim, a script can be used to request significant resources, like logos, of the websites to test. By measuring the time that elapses until the resource is loaded, an attacker can determine whether the victim has already visited a specific page or not. Assuming that websites cause the browser cache to store static content like logos, a short loading time occurs if the requested content was loaded from the browser cache, whereas a longer loading time is caused by loading the file from a web server.

This procedure is also possible without the use of runtime-analysing scripts [10]. To do so, a sequence of requests, two to the own origin and one to a resource of a website to be tested, needs to be defined in a HTML document in immediate succession. The first request to the own origin triggers the start of the time analysis. Next, a resource of a website of interest gets requested like in the procedure that has been described before. As now, there is no application logic in the client's browser that could measure the time elapsing while loading the external resource, a second request to the HTML document's origin gets triggered immediately after the external resource was loaded completely. By analysing the time difference between the incoming requests on the origin server, conclusions about the duration of the loading time for the external resource can be drawn on the server side. The practical implementation of this method requires the consideration of parallel loading of sub-resources. The attacker thus needs to ensure that all requests of a sequence are executed successively after the previous one is completed. This can be archived using the `onload` event of a DOM element. Even though JavaScript is required to implement this, the evaluation takes place on the server side, so no suspicious code is visible to the client.

Using both procedures, it is also possible to obtain geographic information of website visitors. By checking locally related pages it is possible, in particular for a limited number of users to be identified, to infer their location or to identify them on the basis of their location [16].

The third way to implement a tracking mechanism based on the browser cache is utilising HTTP headers that are originally intended to be used for validating cached resources. The HTTP headers `Last-Modified` and `ETag` are suitable to transport user-based identifiers. These are stored together with the corresponding resource to the browser cache. If a user is visiting a website that requires this previously stored resources, the web browser performs a *freshness validation* by transmitting the `Last-Modified` or the `ETag` header values. On this way, the user-based identifier is available on the server side and can be used in order to derive a usage behaviour [17].

4 Browser Cache Experiments

To examine the impact of browser caches with respect to the current state of the web, an extensive analysis is carried out. The top 500 URLs of Tranco [20] retrieved on August 19, 2019 serves as data basis. It results from a combination of well-known website ranking lists like Alexa, Cisco Umbrella, Majestic and Quantcast which has been prepared as a reliable and manipulation-free database for security and privacy research. Using a Chromium devtools extension, each URL of the list gets requested five times in a row with a time interval of 20 s. The recorded traffic is exported in the HTTP Archive (HAR) [23] format for further analysis. Since data which is stored persistently on the client's hard disk is of particular interest, the memory-cache, which is used by Chromium to temporary store content for opened tabs, is cleared before each page call. The records thus can be reduced to the usage of the disk-cache. In case of examinations with activated browser cache, every website is called once before starting the measurements to fill the cache. Since the content and thus also the transmitted data of a website can be time-dependent, the measurement conditions described in the following are carried out in parallel.

4.1 Effects on Network Performance

The first part of the analysis focuses on the effect of a browser cache on the transmitted data volume when a website is accessed. For this purpose, the transferred data volume of incoming responses that do not originate form the disk-cache is computed. In order to avoid measurement errors, only the sub-resources of a website that are present in the data set of all measurement conditions are considered. The median of resulting data volume for the five available measurements is determined and used for further evaluation. This procedure is performed for three test cases:

(a) activated browser cache
(b) deactivated browser cache (via option in the devtools)
(c) browser cache limited to same-origin and a *https* pattern in the URL

Test case (c) was achieved by using a developed browser extension that prevents the browser from loading resources from the cache that do not originate from the host of the page the user is attempting to access. As the affiliation of a resource is determined by its URL, the usage of CDNs that do not include the full name of the host, specified for the main HTML document, leads to an exclusion from the browser cache. In addition, caching is limited to URLs starting with the pattern *https*.

The results show a median of saved data of 95.9% for the comparison of activated and deactivated browser cache (test cases (a) and (b)). As shown in Fig. 1, half of the requested websites reach a data saving of between 89.7% and 98.9% in this measuring condition. Restricting the use of the browser cache to resources from the originally requested origin and a *https* schema in the URL results in a significantly different outcome. As shown in Fig. 2, half of the requested websites archive a data saving of 0.35% to 77.7%. The median reaches a value of 37%.

The comparison of the data savings of the two evaluations clarifies the number of resources that originate from sources that are not initially related to the originally requested website. Some might originate from CDNs that are operated by the provider of the website, but whose URL cannot simply be traced back to the URL of the main HTML document. Another part are third-party resources which are not associated with the originally requested website.

The evaluation of measurement conditions (a) and (b), the comparison of activated and deactivated browser cache, clearly points out the importance of this technology with regard to the scalability of a distributed system like the web. The renouncement of a browser cache thus leads to a drastic increase of required bandwidth, which is unacceptable, especially with mobile applications.

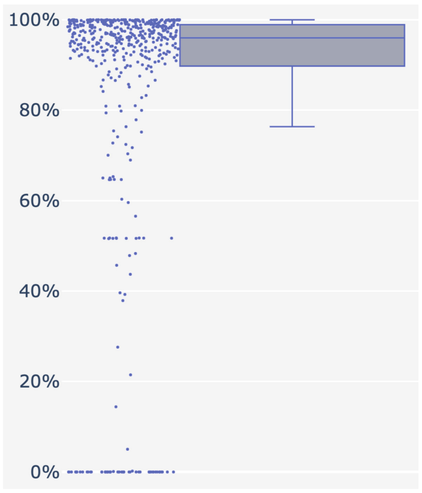


Fig. 1. Percentage of data saving by using the browser cache of Chromium for ubuntu. Median: 95.9%, q1: 89.7%, q3: 98.9%, min: 0%, max: 100%

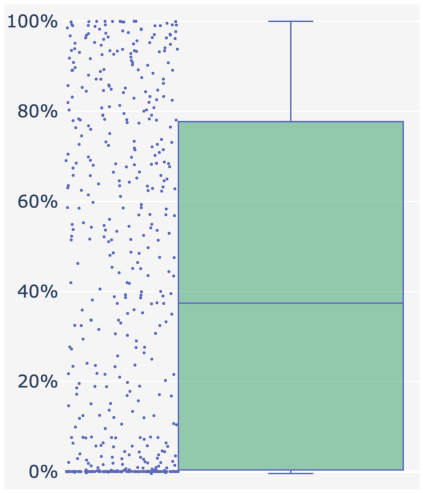


Fig. 2. Percentage of data saving by using the browser cache of Chromium for ubuntu with limitation to same-origin resources and *https* URL patterns. Median: 37%, q1: 0.35%, q2: 77.7%, min: 0%, max: 100%

4.2 Current Security and Privacy Risk Assessment

The second part of the analysis focuses on potential threats caused by the browser cache. Browser Cache Poisoning attacks are particularly dangerous if they are applied on resources that are used by several websites likewise. In order to quantify this danger, the records are analysed to find cached resources that are shared among several websites. We again use the top 500 websites by Tranco [20] and capture the HTTP traffic with activated browser cache. The logs are analysed as follows. If a sub-resource is loaded from the browser cache, the URL

Table 1. The 10 most referenced resources by the Tranco top 500 websites

URL of resource	Web page referencing resource
https://www.google-analytics.com/analytics.js	202 (40.4%)
https://connect.facebook.net/en_US/fbevents.js	111 (22.2%)
https://connect.facebook.net/signals/plugins/inferredEvents.js?v=2.9	111 (22.2%)
https://www.googletagservices.com/tag/js/gpt.js	81 (16.2%)
https://tpc.googlesyndication.com/safeframe/1-0-35/html/container.html	80 (16%)
https://securepubads.g.doubleclick.net/gpt/pubads_impl_2019082201.js	79 (15.8%)
https://www.googletagservices.com/activeview/js/current/osd.js?cb=%2Fr20100101	67 (13.4%)
https://www.googletagservices.com/activeview/js/current/osd_listener.js?cache=r20110914	65 (13%)
https://sb.scorecardresearch.com/beacon.js	64 (12.8%)
https://securepubads.g.doubleclick.net/gpt/pubads_impl_rendering_2019082201.js	60 (12%)

of the initiating website is captured. This leads to a list of web pages loading the same resource from the browser cache.

Table 1 contains the top 10 shared resources, sorted descending by the amount of requesting websites. 40% of the websites requested in this analysis reference the script *analytics.js* delivered by the host *google-analytics* which is located in the browser cache. Compromising this file in would therefore have impact on a large number of websites.

Reviewing Browser Cache Poisoning Attacks. To investigate the relevance of Browser Cache Poisoning attacks on current web browsers, a further experiment was carried out. This draws particular attention to the relationship between the validity of TLS certificates and resources loaded from the browser cache. A similar investigation has been done in [15] in 2015. In order to verify how modern web browsers deal with this kind of attack, a renewed assessment was done. The investigation is based on the assumption that files in the browser cache are assigned solely based on a single cache key i.e. the URL, but no distinction is made as to whether the resource was transferred via a valid TLS connection or not. A valid TLS connection is defined to be based on a certificate whose authenticity is fully and successfully verified by the web browser. The experiment set up (see Fig. 3) is described in the following.

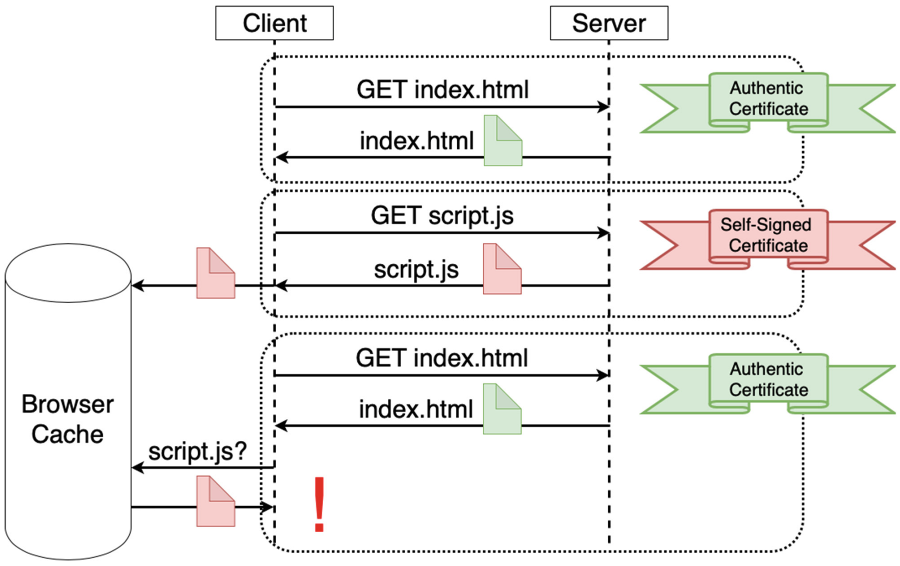


Fig. 3. Procedure of Browser Cache Poisoning attack performed in order to test the relationship between TLS certificates and cached resources

The client visits a website that is providing a valid TLS certificate. This website serves an index HTML file and a JavaScript file as sub-resource. Both files are delivered with cache statements that forbid the browser to load them from the cache. In order to simulate a MITM attack, the delivering web server changes its configuration and uses a self-signed certificate that is not trusted by the browser. The client now reloads the website and receives an TLS warning message. After he successfully clicked through the warning, the web server delivers its data as usual. As before, two resources are transferred, an index HTML page and a JavaScript file. The latter now contains malicious code, but does not provide any caching instructions. After a further change of the configuration, back to the valid TLS certificate, the client updates the requested page again. Although he now uses a trusted TLS connection, the browser loads the manipulated JavaScript file from the browser cache. This will be the case until the user explicitly instructs his web browser to request all resources from the origin, or clears the cache.

To perform this experiment, an nginx web server is used to distribute the HTML and JavaScript file. A Let's Encrypt certificate was issued for the server, in addition a self-signed certificate was created. The server configuration is used to switch between the two certificates, whereby the self-signed certificate simulates a MITM attack. When changing the certificates, also the content of the JavaScript file changes, whose function consists of inserting the currently configured scenario (good certificate/bad certificate) as string into an object of the HTML page. The selected scenario assumes that the HTML file is not cached by the browser in both configurations. In the case of a valid certificate, the JavaScript file should not be cached, but the MITM attack should save it persistently. Figure 4 shows the Cache-Control header configuration defined on the server.

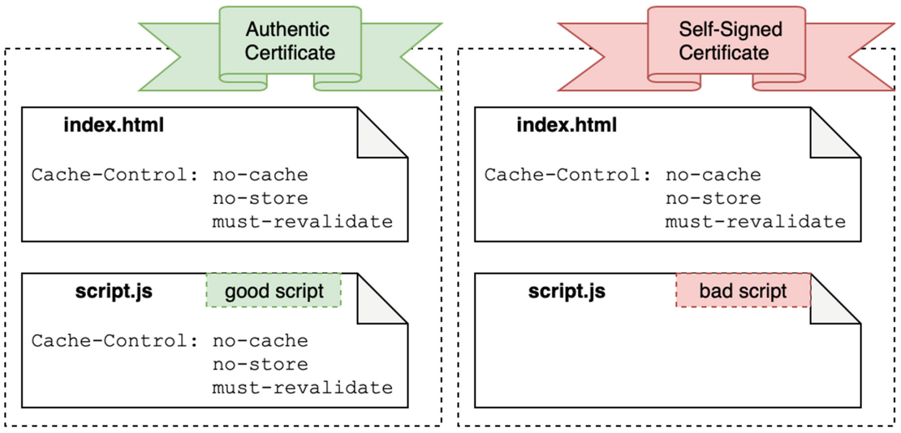

Authentic Certificate
index.html
Cache-Control: no-cache no-store must-revalidate
script.js
good script
Cache-Control: no-cache no-store must-revalidate
Self-Signed Certificate
index.html
Cache-Control: no-cache no-store must-revalidate
script.js
bad script

Fig. 4. Cache-Control header of the resources provided by the server during the simulated Browser Cache Poisoning attack

The results (see Table 2) show that all tested web browsers are vulnerable to the chosen attack. Due to the cache implementation of the Chrome browser, the

JavaScript file delivered with the valid certificate gets cached already, so that an explicit reload of the page need to be done to load the manipulated script. For all browsers, a TLS certificate warning needs to be clicked through before accessing the web site transmitted when using a self-signed TLS certificate. After switching back to the authentic certificate, all browsers load the manipulated script from the browser cache. Chrome, Internet Explorer and Edge also cache the HTML file, which results in the certificate warning in the URL-bar remaining for the time being. Firefox updates the TLS information in the address-bar; Safari does not distinguish between accepted self-signed and authentic certificates. After the web browsers are closed and restarted, Internet Explorer, Edge and Safari still load the manipulated JavaScript file from the browser cache, this time without any certificate warnings. The scenario constructed is probably rare in reality, since the described attack can be prevented by simple mechanisms such as Subresource Integrity [7]. Nevertheless, it shows that none of the browsers tested takes the authenticity of the certificate into account when caching.

Table 2. Impacts of the simulated browser cache poisoning attack

	Chrome v76 (Windows & MacOS)	Firefox v69 (Windows & MacOS)	Internet Explorer v11 (Windows)	Edge v44 (Windows)	Safari v12 (MacOS)
Caches malicious script (self-signed certificate)	(Yes) after forced page reload	Yes	Yes	Yes	Yes
Loads malicious script (valid certificate)	Yes[a]	Yes	Yes[a]	Yes[a]	Yes
Loads malicious script after browser restart (valid certificate)	No	No	Yes	Yes	Yes

[a]As the index.html file is loaded from the cache too, the certificate warning in the URL-bar is still present.

5 User-Centred Mitigation Strategies

Vulnerabilities of browser caches can affect the security and privacy of users. The analyses described in the previous section points out that modern web browsers still not provide sufficient mechanisms to prevent such attacks. The likewise usage of shared resources shown in Table 1 increases the impact of an incident. In order to improve the security of the browser cache without extensive changes in the implementation of web browsers, users could make use of additional tools. In the following, user-centred mitigation strategies will be presented and analysed. As it is well known that security-relevant technologies require a high degree of usability in order to achieve the intended improvements in practice [27], we reviewed the following mechanisms under consideration of efficiency and task adequacy. First, we will highlight the simplest method imaginable, deleting the browser cache. Due to the different implementations of the web browsers, the usability of this function is not sufficient in all cases. Afterwards, available browser extensions are examined which influence the functionality of the browser cache to achieve a higher level of security and privacy.

5.1 How to Clear the Browser Cache

To evaluate the usability of the cache delete function for the web browsers Firefox, Chrome, Safari, Internet Explorer and Edge, the procedure is examined by focusing on the user's effort. Table 3 shows the number of clicks necessary to clear the cache.

The clear cache function of Firefox is located in the security and privacy area of the browser settings. Five clicks are required to use it.

Chrome allows to open the corresponding dialog without navigating to the general browser settings. The deletion of browser data can be limited to a certain period of time. As the last hour is selected as default, users first must adjust the period to the total interval of time. If this is done once, the cache can be cleared within four clicks.

The Safari browser does not offer any option to clear the cache when using the default configuration. With a total of five clicks, the developer settings must be activated first to enable the option to empty the cache memory. If this is done, two clicks are necessary.

To clear the browser cache in Internet Explorer, the appropriated dialog can be accessed using the *Security* sub-menu of the browser's main-menu. Unfortunately it is marked with the term *browser history*. Four clicks are required to clear the cache.

Edge requires a procedure similar to Firefox. The corresponding function can be found in the security and privacy section located in general settings. Five clicks need to be done to clear the cache.

With the exception of Firefox and Edge, the procedures for deleting the browser cache differ. Especially the necessity of activating the developer options on Safari seems unnecessarily complicated. Chrome, Firefox, Internet Explorer and Edge also offer a shortcut to display the dialog box for deleting browser data. In this case, Firefox will open a window which is different from the one accessible via the menu. Similar to Chrome a time period must be selected for which the data should be deleted.

Table 3. Number of clicks required to clear the browser cache

	Firefox	Chrome	Safari	Internet Explorer	Edge
Number of clicks	5	4	5 + 2	4	5

Instead of clearing the browser cache, a private browsing session can be used for all considered web browsers. The resources cached during private sessions get cleared once all private browsing windows are closed.

5.2 Tool-Based Solutions

Frequently clearing the full browser cache does not seem to be an efficient way to deal with the present security and privacy threats when considering the amount of data saved using this technology. A way of defining cache policies for sensitive websites, such as online banking or risky resources like the ones originating from different origins, would be more desirable. Browser add-ons can provide additional functionality that could lead to an improvement for the user's security and privacy. We analysed extensions that are available at Google Chrome Web Store, Firefox Add-ons and GitHub considering a functionality which exceeds clearing the browser cache as well as a usable design which enables ordinary users to make use of them. We used the keyword "cache" to search for browser plugins at Chrome and Firefox extension stores as well as "browser cache extension" to find relevant projects on GitHub. Due to a large number of results, the analysis is restricted to the first 60 hits.

Extensions found at the Chrome App Store can be grouped into three categories. The largest group (26 results) provides only deletion of the cache or in some cases further storages such as the browsing history. Another category (8 results) refers to applications that offer to display cached elements or to render web pages using resources stored in the browser cache. The third category (3 results) contains applications that can manipulate the caching behaviour. These include *NoCache* [24], *PowerCache* [18] and *Supercache* [21]. *NoCache* provides a button to disable the browser cache. Using *PowerCache* it can be individually specified which elements should not be loaded from the browser cache. This is done by using a regex pattern to filter the requested URLs. Requests that are manipulated by the extension are listed in the add-on. This application is primarily intended for developers and thus is not designed to be usable for ordinary web users. *Super Cache* is using a simple and minimalist user interface. One can specify whether stylesheets, images or scripts should be loaded from the cache. The options *Default* (no manipulation), *Cache* or *NoStore* can be set. These settings can be configured for each website individually. As this extension only manipulates the Cache-Control header directive of response headers, it can only prevent resources from being stored to the browser cache. If a file is already located there, it will be used to render the website anyway.

The analysis of Firefox browser extensions was done in the same way. Among the first 60 results, 13 applications can be found which can clear the cache, six applications that can display cache contents and two applications that allow deactivation of the cache (*Toggle Cache* [25] and *Cache It Out* [6]). Five applications provide reloading of a page without using files from the browser cache. As this concept seems to be similar to clearing the cache or performing a full reload of a website, these extensions are neglected. Due to a restriction of the implementation of the extension API used by Firefox, it is not possible to control the behaviour of the browser cache in detail. This would require the manipulation of a HTTP request with regard to the caching behaviour before it is processed by the browser, which is not possible in Firefox extensions.

The search on GitHub revealed eleven applications that allow to clear the browser cache. Four projects offer the possibility to display content from the cache. Only one application (Opera-disable-cache [9]) can deactivate the browser cache. As it is implemented for the Opera browser, it will not be discussed further. Table 4 provides an overview of the features of the remaining five applications that can control the caching behaviour.

Table 4. Comparison of five browser extensions that allow to control caching behaviour

	NoCache (Chrome)	PowerCache (Chrome)	SuperCache (Chrome)	Toggle Cache (Firefox)	Cache It Out (Firefox)
GUI	X	✓	✓	X	X
Page specific settings	X	X	✓	X	X
Resource-type specific settings	X	✓	✓	X	X
View cached resources	X	✓	X	X	X
Multiple settings per website	X	X	✓	X	X

(✓: feature provided, X: feature not provided)

6 Discussion

The security and privacy risks presented in this paper, as well as the results of the experiments carried out, demonstrate that modern web browsers are still providing insufficient precautions to protect users. The analysis of present user-centred mitigation strategies points out that further work needs to be done in order to compensate the shortcomings of current browser cache implementations. The following section should take up the most important issues and give recommendations on how these can be solved effectively.

Web Browser Extensions. In order to improve the browser cache with regard to security and privacy, it would be conceivable to develop browser extensions that control the behaviour of the cache according to users' needs. Using browser add-ons, shortcomings of the current caching procedure could be diminished without depending on extensive and time-consuming developments of improved web browsers. When looking at the APIs for Chrome and Firefox extensions, we noticed that there is no direct way to manipulate the behaviour of the browser cache [1,5]. By setting the Cache-Control header directive `no-cache` in the request header, it is possible to prevent Chrome from loading a resource from the cache. The documentation of Chrome contradicts this observation, but describes this information as unstable. [2]. It would be desirable if browser vendors would provide well-documented interfaces by which developers can create high-performance solutions that give users more control on the behaviour of their web browsers.

Tracking via Browser Cache. As discussed in Sect. 3, the browser cache can be utilised to attack users' privacy. By using individual identifiers, a web tracking mechanism can be implemented, very similar to the functionality of cookies. We have not yet been able to establish any more detailed investigations into this matter, but consider this to be absolutely necessary. While the introduction of the General Data Protection Regulation (GDPR) increased users' awareness on privacy related risks of cookies due to the omnipresence of cookie info boxes on many websites, the risks related to the browser cache are far less present and seem to have received little attention so far. We plead for a comprehensive analysis of possible further uses of the browser cache with regard to the risks for the privacy of users. This includes but is not limited to a critical consideration of *freshness validation* mechanisms as they can be misused to exchange user identifiers using `ETag` or `Last-Modified` header values.

Cache Keys and TLS Certificates. Restricting the cache key to the URL for resources transferred over TLS secured connections can lead to urgent security risks. However, this practice seems incomprehensible, since information about the authenticity of the certificate used in the TLS connection is available in the web browser. Resources transferred via TLS connections based on different certificates should not be considered identical. In fact, this contradicts the security service of authenticity. A resource should therefore be bound to the certificate used during the transfer. Bypassing the browser cache for unauthentic TLS connections could prevent Browser Cache Poisoning attacks by limiting the impact of MITM attacks to the moment of the actual attack. Investigations also revealed that the devtools of the Firefox web browser, unlike the information in the URL-bar, do not distinguish between authenticated and, for example, self-signed certificates. In both cases a green icon appears beside the transferred resources displayed in the network tab. This does not seem to be sufficiently differentiated and could lead to confusion among developers.

Browser Cache Partitioning. To prevent attacks that could harm the privacy of users, Chrome plans to partition the cache [4]. This is to be achieved by using the origin URL of the website requested by the user as an additional key for a cached resource. This approach is to be welcomed as it can prevent attacks like CTAs. Developers of Firefox are also considering this improvement [3]. It remains to be seen whether this gets actually implemented.

7 Conclusion

HTTP browser caches are important components of distributed systems like the web as they enable these systems to scale at large. We show that half of the 500 most popular websites from the Tranco list archive savings between 89% and 99% of the data volume to be transmitted when browser caching is enabled. Thus, disabling the browser cache has severe performance and economic consequences for service providers and end users respectively.

At the same time though, browser caches can be misused to perform user tracking and to persistently store malicious code on the client's computer. The current practice does not sufficiently balance security and performance trade-offs. By taking a closer look at the files stored inside the cache we found resources that are used by up to 40% of the examined websites. As these resources most commonly stem from third parties providing analytics or advertising services, the potential for user tracking is inherent. Moreover, we found that modern web browsers still do not distinguish between valid and self-signed TLS certificates when storing and reusing resources from the browser cache. Although this weakness opens the door for Browser Cache Poisoning attacks and has been known since 2015 [15], it is still exploitable in the most current versions of major browsers.

Hence, providing a more fine-grained control of the cache policy enforced by the browser is essential. Browsers themselves offer caching by default and simple settings to clear the cache. In some cases these settings are even well hidden from the end user. Some browser extensions aim at closing this gap with more enhanced features. However, our analysis of these extensions emphasises that they provide only very fragmented functionalities with most of them only going slightly beyond browsers' builtin functionalities.

Overall, we argue that more research is required in respect to browser caches and their relation to the security and privacy of end users. This would enable browser vendors to close existing vulnerabilities and provide more elaborated cache control settings for end users as well as APIs for developers and researchers. The latter will foster the iterations towards novel approaches to balance performance and security.

References

1. Chrome APIs - Google Chrome. https://developer.chrome.com/extensions/api_index. Accessed 05 Sept 2019
2. chrome.webRequest - Google Chrome. https://developer.chrome.com/extensions/webRequest. Accessed 05 Sept 2019
3. Double-keyed HTTP cache Issue #904 whatwg/fetch. https://github.com/whatwg/fetch/issues/904. Accessed 05 Sept 2019
4. Partition the HTTP Cache - Chrome Platform Status. https://www.chromestatus.com/feature/5730772021411840. Accessed Sept 05 2019
5. WebExtensions. https://developer.mozilla.org/de/docs/Mozilla/Add-ons/WebExtensions. Accessed 05 Sept 2019
6. Firefox user 13863091: Cache it out. https://addons.mozilla.org/en-US/firefox/addon/cache-it-out/. Accessed 05 Sept 2019
7. Akhawe, D., Braun, F., Marier, F., Weinberger, J.: Subresource Integrity. W3c Reccomendation, W3C (2016). https://www.w3.org/TR/SRI/. Accessed 05 Sept 2019
8. Bansal, C., Preibusch, S., Milic-Frayling, N.: Cache timing attacks revisited: efficient and repeatable browser history, OS and network sniffing. In: Federrath, H., Gollmann, D. (eds.) SEC 2015. IAICT, vol. 455, pp. 97–111. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18467-8_7

9. Digital, b.: Opera extension to disable browser cache, perfect for developers: biati-digital/opera-disable-cache. https://github.com/biati-digital/Opera-disable-cache. Accessed 05 Sept 2019
10. Felten, E.W., Schneider, M.A.: Timing attacks on web privacy. In: Proceedings of the 7th ACM Conference on Computer and Communications Security, CCS 2000, Athens, Greece, pp. 25–32. ACM, New York (2000). https://doi.org/10.1145/352600.352606
11. Fielding, M.N.R., Reschke, J.: RFC 7234: hypertext transfer protocol (HTTP/1.1): caching. Technical report RFC 7234, IETF (2014)
12. Fielding, R., et al.: RFC 2616: hypertext transfer protocol-(HTTP/1.1). Technical report RFC 2616, IETF (1999)
13. Fielding, R., Reschke, J.: RFC 7232: hypertext transfer protocol (HTTP/1.1): conditional requests. Technical report RFC 7232, IETF (2014)
14. Fleischer, G.: Implementing web tracking. In: Black Hat USA 2012 Conference Briefings, pp. 1–37 (2012)
15. Jia, Y., Chen, Y., Dong, X., Saxena, P., Mao, J., Liang, Z.: Man-in-the-browser-cache: persisting HTTPS attacks via browser cache poisoning. Comput. Secur. **55**, 62–80 (2015). https://doi.org/10.1016/j.cose.2015.07.004
16. Jia, Y., Dong, X., Liang, Z., Saxena, P.: I know where you've been: geo-inference attacks via the browser cache. IEEE Internet Comput. **19**(1), 44–53 (2015). https://doi.org/10.1109/MIC.2014.103
17. Juels, A., Jakobsson, M., Jagatic, T.N.: Cache cookies for browser authentication. In: 2006 IEEE Symposium on Security and Privacy (S P 2006), pp. 5–305, May 2006. https://doi.org/10.1109/SP.2006.8
18. Khachatryan, A.: Power cache. https://chrome.google.com/webstore/detail/power-cache/famkodflhompmapangljedfdcfeligih?hl=de. Accessed 05 Sept 2019
19. Kuppan, L.: Attacking with HTML5 (2010). https://media.blackhat.com/bh-ad-10/Kuppan/Blackhat-AD-2010-Kuppan-Attacking-with-HTML5-wp.pdf. Accessed 05 Sept 2019
20. Le Pochat, V., Van Goethem, T., Tajalizadehkhoob, S., Korczynski, M., Joosen, W.: Tranco: a research-oriented top sites ranking hardened against manipulation. In: Proceedings 2019 Network and Distributed System Security Symposium, San Diego, CA. Internet Society (2019). https://doi.org/10.14722/ndss.2019.23386
21. Mathur, T.: Super-cache. https://chrome.google.com/webstore/detail/super-cache/fglobbnbihckpkodmeefhagijjcjnbeh?hl=de. Accessed 05 Sept 2019
22. Nguyen, H.V., Lo Iacono, L., Federrath, H.: Systematic analysis of web browser caches. In: Proceedings of the 2nd International Conference on Web Studies, WS.2 2018, Paris, France, pp. 64–71. ACM, New York (2018). https://doi.org/10.1145/3240431.3240443
23. Odvarko, J., Jain, A., Davies, A.: HTTP Archive (HAR) format (2019). https://w3c.github.io/web-performance/specs/HAR/Overview.html Accessed 05 Sept 2019
24. Oluwaseye: No cache. https://chrome.google.com/webstore/detail/no-cache/hckocmggmdfdnjjomghmhllibmdobdll. Accessed 05 Sept 2019
25. Reimer, M.: Toggle cache. https://addons.mozilla.org/de/firefox/addon/togglecache/?src=search. Accessed 05 Sept 2019

26. Saltzman, R., Sharabani, A.: Active man in the middle attacks. OWASP AU (2009). http://www.security-science.com/pdf/active-man-in-the-middle.pdf. Accessed 05 Sept 2019
27. Whitten, A., Tygar, J.D.: Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In: Proceedings of the 8th Conference on USENIX Security Symposium, vol. 8, SSYM 1999, Washington, D.C., pp. 14–14. USENIX Association, Berkeley (1999). http://dl.acm.org/citation.cfm?id=1251421.1251435

Improving Privacy Through Fast Passive Wi-Fi Scanning

Frederik Goovaerts[1], Gunes Acar[2], Rafael Galvez[2], Frank Piessens[1], and Mathy Vanhoef[1,3](✉)

[1] imec-DistriNet, KU Leuven, Leuven, Belgium
Mathy.Vanhoef@cs.kuleuven.be
[2] imec-COSIC, KU Leuven, Leuven, Belgium
[3] New York University Abu Dhabi, Abu Dhabi, United Arab Emirates

Abstract. Traditionally, Wi-Fi networks are discovered by actively transmitting probe requests. The alternative, passive scanning, is rarely used because it is substantially slower. Unfortunately, active scanning can be abused to track users based on (physical) fingerprints of probe requests. Previous work attempted to address these issues by making active scanning more privacy-friendly. For instance, Franklin et al. proposed to make implementations more uniform (USENIX Security 2006), and Lindqvist et al. suggested to use encrypted probe requests (WiSec 2009). However, a better approach is to make passive scanning faster. This motivates vendors to use passive scanning, increasing the privacy of users.

Motivated by the above insight, we improve the performance of passive scanning. We implement our proposals on Android, and show the average time needed to connect to a known network using passive scanning now matches active scanning. Additionally, we implement a new network-discovery mechanism that drastically decreases scanning times, and present a new method to fingerprint Wi-Fi radios. All combined, our results show that passive scanning is a viable and more privacy-friendly alternative to active scanning.

Keywords: Tracking · Anonymity · Passive scanning · Priority scanning

1 Introduction

Practically all mobile devices discover nearby Wi-Fi networks by actively sending probe requests [10]. Unfortunately, properties of the physical Wi-Fi signal of probe requests can be abused to fingerprint and track devices [4]. These physical properties of the Wi-Fi signal are caused by unique imperfections in each radio transmitter. As a result, whenever a probe request is sent, no matter what its content, it becomes possible to track the sender. Additionally, probe requests may contain the unique MAC address of the transmitter, and can contain other sensitive information [10,21]. We find that in practice people are indeed being tracked based on the Wi-Fi signals of their devices. For example, garbage bins tracked people in London based on probe requests [5].

Previous works tackled this issue by trying to make active scanning more privacy-friendly. For instance, Franklin et al. proposed to make active scanning implementations more uniform [9, §7], Greenstein proposed to encrypt probe requests [11], and Lindqvist et al. suggested a different protocol to encrypted probe requests [16]. More recently, Vanhoef et al. advised to simplify and unify the content of probe requests [23], and Matte et al. suggested to randomize the timings of probe requests [18]. However, our position is that trying to improve active scanning is not the way forward. We believe this because new tracking techniques against active scanning are inevitable, since in general *any* frame being transmitted can be fingerprinted based on physical-layer properties [4]. To defend against this, we should be making passive scanning more performant, such that it becomes a viable alternative to active scanning.

Under passive scanning, a device does not send any frames to discovery networks. Instead, nearby Wi-Fi networks are instead detected by listening for beacon frames that all APs periodically transmit. These beacon frames contain the same information as probe responses, and hence allow a client to determine all relevant parameters of the network. This makes it impossible to track users based on probe requests. Unfortunately, having to wait for beacons on each channel makes passive scanning significantly slower than active scanning, and therefore it is rarely used. We overcome this obstacle by increasing the performance of passive Wi-Fi scanning.

Apart from privacy pitfalls, active scanning also reduces the available bandwidth. That is, the airtime consumed by probe requests and responses can be quite large. For example, in crowded places they take up more than 10% of available airtime [14], and can reduce the throughput of clients by 17%. Hence, apart for privacy advantages, passive scanning would also free up airtime.

Inspired by the privacy and bandwidth benefits of passive scanning, our goal is to increase its speed, such that it becomes a viable alternative to active scanning. This will incentivise vendors to use passive scanning instead, thereby eliminating the privacy downsides of active scanning. To avoid the longer scanning durations under passive scanning, we introduce the concept of priority scans. Under a priority scan, a set of priority channels is scanned first, and networks detected on these channels are returned immediately. We implement these modification on Android to evaluate our techniques in practice. Additionally, we propose a novel scanning technique where networks advertise all neighboring networks that they can detect, and present a new method to fingerprint Wi-Fi radios. The client can then use this information to drastically reduce the average scanning duration.

To summarize, our main contributions are:

- We present a novel method to fingerprint of type of Wi-Fi radio that a device uses (Sect. 3).
- We improve passive scanning to reduce the time needed to discover and connect to (known) networks (Sect. 4).

– We implement and test our proposal on Android (Sect. 5).
– We propose to advertise neighboring networks to drastically improve the discovery time of known networks (Sect. 6).

Finally, we discuss related work in Sect. 7, and conclude in Sect. 8.

2 Background

In this section we introduce relevant parts of the 802.11 standard, and we discuss our threat model.

2.1 Network Discovery

An essential task of a wireless devices is discovering nearby networks. The 802.11 standard provides two methods to discover nearby Wi-Fi networks. The first is passive scanning, and the second is active scanning. We briefly introduce both:

Passive Scanning. All Wi-Fi networks periodically transmit beacon frames. These beacons are used to announce the presence of a network, to synchronize clocks between associated clients, and so on. Since they are periodically broadcasted, clients can passively listen for them to detect nearby networks. Beacons include all network configuration parameters that must be obtained before connecting to a network. In particular, beacons contain the Service Set Identifier (SSID), supported data rates, supported or required security protocols, and so on. By default, APs transmit a beacon every 102.4 ms, though this can be configured differently. Because clients must search for networks on all channels, this makes the default passive scan slow and therefore rarely used.

Some APs can be configured to exclude the SSID in beacons. These are called hidden networks. Doing this has the advantage that nearby clients do not show the SSID (i.e. the network name) in their user interface. To detect hidden networks, active scanning must be used and the SSID of the hidden network must be included in all probe requests. Because including the SSID in probe requests can leak sensitive information about the user, the usage of hidden networks is no longer a recommended practice [10,19,23].

Active Scanning. With active scanning, the client transmits broadcast probe requests, and in response nearby APs reply with probe responses. These probe responses contain all the information required to connect with a network. As a result, a device can quickly detect nearby networks using active scanning.

Unfortunately, probe requests may contain a significant amount of information about the client's device. For instance, if MAC address randomization is not used, probe requests contain the permanent MAC address of the client. This can be used to trivially track users [3]. Moreover, properties of the physical Wi-Fi signal can also be used to fingerprint and track a device [4]. All combined, whenever a client sends probe requests, it becomes possible to track the user.

Finally, active scanning is the only mechanism able of detecting hidden networks. This is because to detect a hidden network, the client must send a directed probe request that contains the SSID of the particular hidden network. If the hidden network is nearby, it will reply with an (ordinary) probe response.

2.2 Frequency Bands and Channels

A Wi-Fi network can operate in various frequency bands, and within one band can operate on several possible channels. The most common frequency bands are the 2.4 GHz and 5 GHz bands. Here, the 2.4 GHz band has 13 channels (in Europe), and out of these 13 channels there are three non-overlapping ones (channel 1, 6, and 11). It is common practice to only use one of these non-overlapping channels, since this avoids cross-channel interference [8].

In the 5 GHz band, there are more than 20 non-overlapping channels. In most regulatory domains (i.e. countries), only a few of these channels can be used freely. All the other channels in the 5 GHz band can only be used if the device supports Dynamic Frequency Selection (DFS). We will refer to these as DFS channels. DFS is a mechanism to avoid interference with radar systems. Among other things, DFS prohibits a device from transmitting until it has listened on the channel fore more than one minute without detecting radar pulses. This means clients are not allowed to immediately send probe requests on DFS channels. Instead, only passive scanning can be used to detect APs on DFS channels.

2.3 Threat Model

The adversary we consider aims to identify and track devices when they are not connected to an AP. We assume the adversary controls enough APs, so any probe request that the victim sends will be captured. Additionally, as shown in [4], we must assume the adversary can fingerprint probe requests based on physical properties of any transmitted frame. Put differently, *any* frame sent by a device can be used to identify and track it [4].

3 Channel Switch Fingerprinting

In this section we describe a novel method to fingerprint devices. We show how this method allows an adversary to differentiate devices based on the type of internal Wi-Fi radio that a device uses.

3.1 Channel Switch Time

When a device is scanning for nearby Wi-Fi networks, it sends one or more probe requests on all non-DFS channels. This means that during a network scan, the Wi-Fi radio is constantly switching channels. Our insight is that the specific type of Wi-Fi radio being used influences how much time it takes to switch from one Wi-Fi channel to another. As a result, if an adversary is capable of accurately

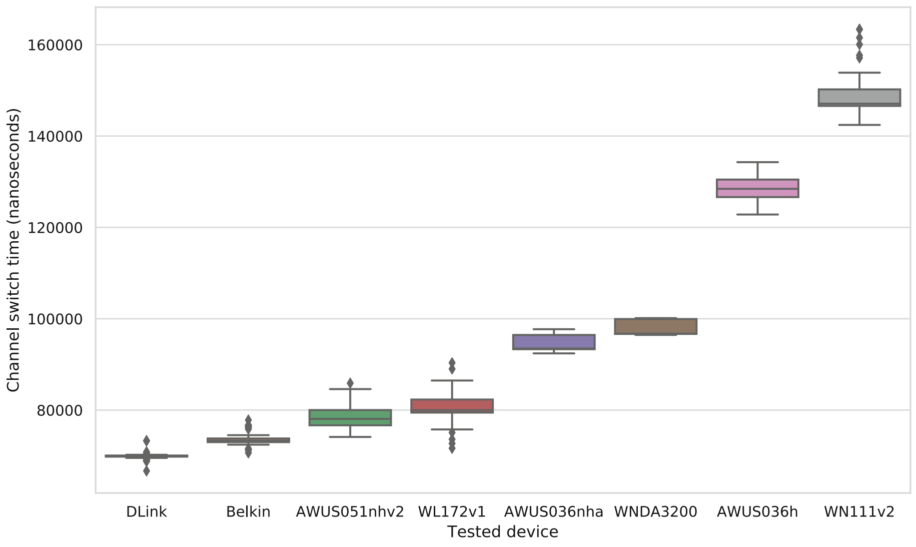


Fig. 1. Box plot of the time between two sequential probe requests sent on different channels, for various types of Wi-Fi radios.

measuring the channel switch time, this information can be used to fingerprint the type of Wi-Fi chip being used.

In practice, most devices will use the same (random) MAC address during one network scan over all channels [17,18,23]. In other words, the same (random) MAC address is used to send probe requests over different Wi-Fi channels. This allows an adversary to measure the time between probe requests sent on different channels. Moreover, this time difference can be measured with high accuracy by relying on hardware receive timestamps of commodity Wi-Fi radios. Due to the capture effect, if the victim device is close to the adversary, it is even possible to use a single Wi-Fi radio to receive frames on two adjacent channels. Since multiple channels are scanned in one individual network scan, the adversary can make multiple channel switch timing measurements. This means the average channel switch time can be calculated, reducing the impact of (temporal) noise. All combined, this means commodity Wi-Fi devices can be used to measure the time between probe requests on different channels.

3.2 Experiments

We measured the channel switch times of 8 USB Wi-Fi radios. The results of these measurements are shown in Fig. 1. We used an Intel Wireless 8265 card to perform the timing measurements. The time difference between two probe requests is calculated based on the hardware receive timestamps of the frames.

From Fig. 1 we learn that different types of Wi-Fi radios result in a different average channel switch time. By calculating the average channel switch time over several Wi-Fi channels, this allows us to accurately differentiate different types of Wi-Fi radios.

3.3 Countermeasures

One possible defense against our novel fingerprinting technique is to use a new random MAC address on each Wi-Fi channel being scanned. This makes it harder to measure the channel switch time. However, this would still allow an adversary to fingerprint physical properties of the Wi-Fi signal, and hence does not deter more powerful adversaries [4]. Instead, in the remainder of the paper, we improve the performance of passive Wi-Fi scanning, such that it becomes a viable alternative to active scanning.

4 Passive Scanning Improvements

In this section we present several techniques to improve the speed of passive scanning. We also define metrics to rigorously evaluate our proposed techniques.

4.1 Proposed Scanning Modifications

In this section, we limit ourselves to backwards-compatible modifications on the client. This makes it easier to deploy our proposals, since no network infrastructure needs to be modified. Additionally, this assures that all existing networks remain discoverable. More concretely we propose the following modifications and optimizations:

Dwell Time Variation. We first evaluate the impact of the dwell time. The dwell time denotes the total time we listen on each channel for beacon frames. This parameter heavily influences the total duration of a passive scan.

Incremental Scanning. In our second modification, partial scanning results are returned to the Operating System (OS) while the scan is still in progress. We call this incremental scanning. More precisely, after scanning each channel, newly discovered APs are immediately returned to the OS. While reporting these results, the Wi-Fi chip continues scanning the remaining channels. The OS can abort the scan once a known network has been discovered.

Static Priority Scanning. This modification is an improvement of the incremental scanning procedure, where we only return discovered APs to the OS after scanning several channels. This removes possible overhead caused by constantly communicating with the OS. At the same time, we prioritize certain channels and scan them first. In particular, we first scan the non-overlapping channels in the 2.4 GHz band (i.e. channel 1, 6, and 11) in a so-called priority scan. After scanning these priority channels, a second scan is performed over the remaining channels. We also examined the 5 GHz band, and observed that channels 36, 40, and 44 are used the most. Hence, in a second variation of static priority scanning, we also include these 5 GHz channels in the (initial) priority scan.

Dynamic Priority Scanning. In our last modification, we dynamically determine the set of priority channels that are scanned first. The specific algorithm that is used to select these priority channels is out of scope for this paper. Instead, we refer to related work for algorithms that determine which networks are likely nearby. For example, the presence of nearby networks can be predicted based on one's location [20], the current day and time [22], and so on.

4.2 Metrics for Evaluation

We now propose several metrics that we will use to quantify the performance of our scanning procedures:

Scan Duration. The scan duration is the total time it takes between listening on the first channel, and the moment collected results are returned to the OS. When using priority scanning, we will refer to the duration of scanning all channels as the full scan time, and the time it takes to scan only the priority channels as the priority scan duration.

Time-to-Connect. The time-to-connect metric measures the time between listening on the first channel, and the moment when the device discovers a known AP. This metric reflects the waiting time that users experience.

AP Discovery Rate. We define the AP discovery rate as the amount of APs discovered relative to the total amount of APs discoverable by the device. This metric needs a reference of all available APs in the vicinity to compare a scan result to. In general, there will be a trade-off between discovery rate and scan duration. Similarly, a longer scan time likely results in a better AP discovery rate, but may negatively impact the time-to-connect metric.

Specific AP Discovery Rate. This metric measures the rate at which a specific set of APs is discovered, relative to the total amount of scans performed. We will use this metric when a known network is nearby, to measure how frequently the network will be detected by various scanning procedures.

4.3 Experimental Setup

We implemented our proposed modifications by modifying the user-space Wi-Fi client. In particular, we modified wpa_supplicant. One advantage of this approach is that every Linux and Android device can then be tested with our modifications.

In our specific setup, we implemented our modifications on Android 7.1.1 AOSP, and used a Google Nexus 5X. We changed the `gPassiveMinChannelTime` parameter (and the analogous `Max` parameter) of the Wi-Fi driver to control the dwell time. Fortunately, this did not require any changes to the driver code.

Instead, we can modify the driver configuration file that contains these parameters using a user-space script[1].

Our incremental and priority scanning procedures are implemented by modifying wpa_supplicant.[2] This means our modifications can be tested on both Android and Linux, making our results easier to replicate. We implemented incremental scanning by issuing separate scan requests for every individual channel. Although this adds a overhead when communicating with the kernel to initiate each scan, part of this overhead is unavoidable. That is, even if we modified the driver or firmware, we still need to send individual notifications to wpa_supplicant after scanning each channel. When measuring the total scan duration of an incremental scan, we add all the individual scan durations, and exclude elapsed time between individual scans. Note that this measurement still includes the overhead caused by communicating with the kernel for initiating each scan. For priority scanning, we also modified wpa_supplicant to issue two individual scans. The first scan covers the priority channels, and the second scan covers all remaining channels.

5 Experimental Results

In this section we experimentally analyze the performance of our scanning strategies. This shows passive scanning can be a viable alternative to active scanning.

5.1 Dwell Time Variation

We first analyzed the impact of varying the dwell time when performing a standard passive scan. Interestingly, the total passive scan duration with a dwell time of 100 ms is only slightly longer than the scan duration of a default active scan. This is because most channels in the 5 GHz must be scanned passively, to avoid interfering with other devices such as weather radars [13, §11.1.4.1]. As a result, even when using default active scanning, these channels are still scanned passively. This implies that, when the 5 GHz band is also scanned, there is only a minor slowdown in scanning duration when switching from the default active scanning procedure to passive scanning.

We also measured the influence of the dwell time on the AP discovery rate. This was done at two locations. First at our office, where there were 30 discoverable APs (see Fig. 2a). Then at a busy international train station where there were 419 discoverable APs (see Fig. 2b). Note that in these figures, the AP discovery rate of a normal passive scan is identical to the discovery rate of a full priority scan. As expected, higher dwell times result in more APs being discovered. With a dwell time of 100 ms, passive scanning matches the discovery rate of default active scanning. For larger dwell times we observed only a slight

[1] This is the file /system/etc/firmware/wlan/qca_cld/WCNSS_qcom_cfg.ini.

[2] Our code, including a build for the Nexus 5X, is available at https://github.com/vanhoefm/nordsec-passivescan.

increase in the discovery rate. With a dwell time lower than 100 ms, the discovery rate quickly drops, with the discovery rate at 50 ms being roughly half of active scanning. We conclude that the optimal dwell time for passive scanning must be at least 100 ms.

5.2 Incremental Scanning

We found that with incremental scanning, the scanning time is slightly higher compared to a standard passive scan. More precisely, the overhead of constantly issuing new scan requests for each channel, reporting the results back to the Operating System, and preparing a new scan, constitutes a 10% overhead. Due to this overhead, we do not consider incremental scanning as a good candidate to replace a default active scan, and will not consider it any further.

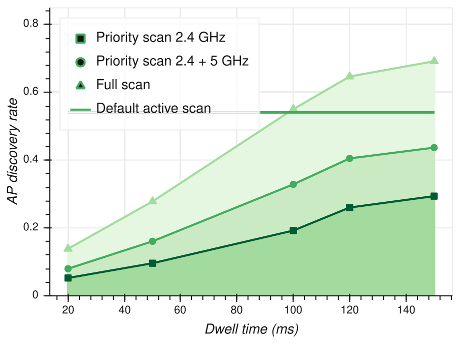
5.3 Static Priority Scanning

To evaluate static priority scanning, we first use the 2.4 GHz channels 1, 6, and 11 as static priority channels. Then we included the 5 GHz channels 36, 40, and 44 as well. We evaluated both the scan duration and AP discovery rate:

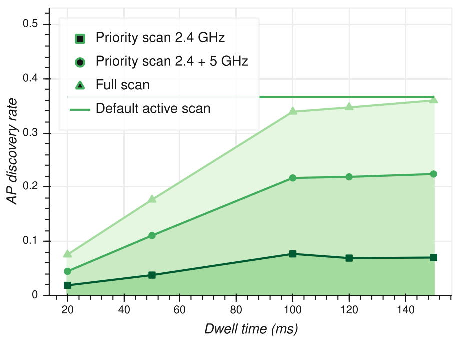
Scan Duration. We found that the ratio of the priority scan duration, compared to a full scan, approximates the ratio of the scanned channels in the priority scan to the total scanned channels. Note that the full scan duration is the same as a standard passive scan duration. For a small set of priority channels, the priority scan is significantly faster than a default scan. That is, when only including three 2.4 GHz channels, the priority scan takes only 8% of the time. When also including the three selected 5 GHz channels, a priority scan takes 15% of the time of a full scan.

AP Discovery Rate. Figure 2a and b contain the AP discovery rate when using priority scanning at our office and the train station, respectively. The lower AP discovery rate around a 120 ms dwell time at the train station (Fig. 2b) is because we had to slightly change our location during the experiment to accommodate passengers. Of interest is how many networks are discovered in a priority scan, compared to a full scan. With the 2.4 GHz priority channel set, 42% of networks are discovered at the office during the priority scan (compared to a full scan), while 22% were discovered at the train station during the priority scan. When also including the 5 GHz priority channels, 61% of networks are discovered at the office compared to a full scan, and 62% are discovered at the train station. This shows that including 5 GHz channels in the priority scan is essential to obtain a high AP discovery rate.

We conclude that, compared to a full passive scan, a priority scan takes 15% of the time, while already discovering close to two-thirds of networks. This shows that priority scanning is a very promising technique. Moreover, a priority scan takes only a fraction of the time, even when compared to the default active scan.



(a) Our office.



(b) International train station.

Fig. 2. The AP discovery rate for various passive scanning procedures at (a) our office; and (b) an international train station. The horizontal line denotes the average discovery rate for a default active scan.

5.4 Dynamic Priority Channels

To test dynamic priority scanning, we assume the client has one known network, and that it treats the (previous) channel of this network as the single dynamic priority channel. We measure the discovery rate and time-to-connect:

Specific AP Discovery Rate. For this experiment we used an environment with little interference and close proximity to the AP. We first tested the specific AP discovery rate for a default active scan, and found that the AP is always detected in the first scan (see Fig. 3). For our passive implementation, we get comparable results. More precisely, with a dwell time of 100 ms, the discovery rate is around 90%. With a dwell time of 120 ms or higher, the AP is always detected. Based on these results, we conclude that passive scanning with a dwell time of 120 ms or higher matches the performance of active scanning. This again shows that passive scanning can form a practical alternative to active scanning.

Time-to-Connect. We investigate the time-to-connect for dynamic priority scanning, and compare it to the default active scanning procedure. For both procedures, we ran the time-to-connect experiment exactly 50 times. Figure 4a shows the resulting time-to-connect histogram for the default active scanning experiments. The average time-to-connect is between 3.8 and 3.9 s, indicating that the AP was always discovered during the first scan. We conjecture that the two separate groups around 3.8 and 3.9 s are caused by internal timing constraints in the Wi-Fi chip.

For passive scanning, the time-to-connect heavily depends on the dwell time. In particular, for a dwell time of 100 ms or lower, multiple scan are sometimes needed to discover the AP (see Fig. 4b). This significantly increases the average time-to-connect, even though most of the time the network is discovered in the

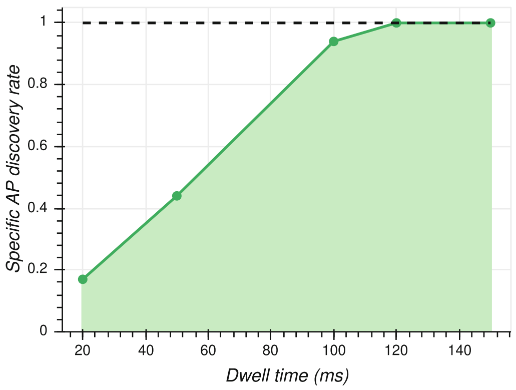
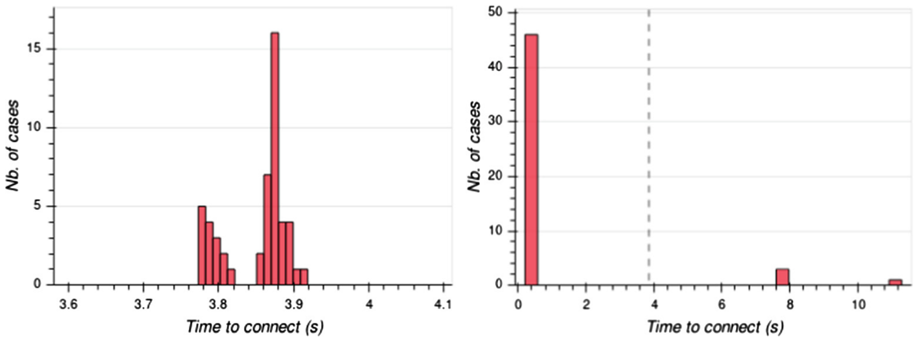


Fig. 3. Specific AP discovery rate for one known network when using passive dynamic priority scanning. The dashed horizontal line marks the average specific AP discovery rate for the default active scanning implementation.

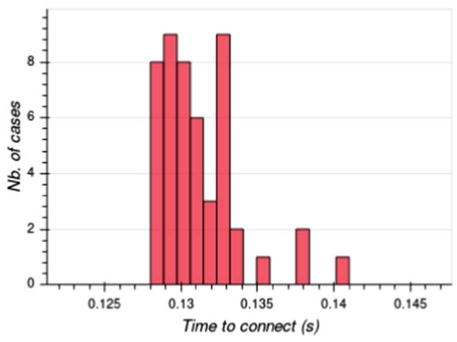


(a) Default active scan.

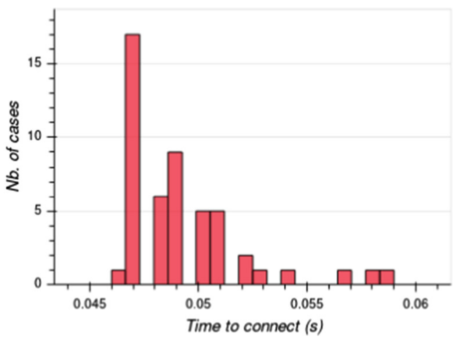
(b) Passive dynamic scan.

Fig. 4. Time-to-connect for (a) default active scanning; and (b) passive dynamic scanning with one priority channel and 100 ms dwelltime. The vertical line denotes the average time-to-connect for the default active scanning procedure.

first scan (see Table 1). However, when using a dwell time of 120 ms or higher, we always discover the AP during the first priority scan. Because this is a priority scan, results of the scan are returned nearly instantly, resulting in a very low average time-to-connect. Notice that with a 150 ms dwell time, the AP is again always discovered in the first priority scan. However, the higher dwell time results in a higher scan duration, and hence also in a higher time-to-connect. Finally, the average time-to-connect for active dynamic priority scanning is 0.049 s. Although this is faster than passive scanning, this difference is likely not noticeable to users, meaning passive scanning remains a viable alternative to active scanning.



(a) Passive dynamic scanning with one priority channel and 120 ms dwelltime.



(b) Active dynamic scanning with one priority channel.

Fig. 5. Time-to-connect for passive and active scanning.

Table 1. Time-to-connect for passive dynamic priority scans

Dwell time (ms)	50	100	120	150
Average TTC (s)	5.62	0.91	0.13	0.16

We conclude that a dwell time of 120 ms minimizes the average time-to-connect. Moreover, compared to the default active scanning procedure, it results in a lower (i.e. faster) average time-to-connect, while being more privacy-friendly.

6 Advertising Neighboring Networks

In this section we propose a novel scanning technique, where APs include basic information of neighboring networks into their beacon frames. We show that providing this information drastically reduces the average scanning time.

6.1 Advertising Neighboring Networks

Many APs are capable of detecting nearby networks on different channels, while at the same time providing normal connectivity to clients. This is commonly used to let the AP automatically operate or switch to the least-used channel, and in practice this feature is often called Dynamic Channel Selection (DCS) or Dynamic Channel Assignment (DCA) [7,24]. More importantly for us, this means many APs are capable of scanning for nearby networks without interfering with normal operations.

To reduce the average scanning time of clients, we propose that every AP advertises all neighboring networks in its beacon frames. In practice, a simplified version of the Neighbor Report element can be used for this [13, §9.4.2.37]. In this element the SSID and channel of neighboring networks is included. In case there are many neighboring networks, and there is insufficient space to advertise them all, then only networks with a high signal strength can be included.

When a client is scanning for networks, it can use the neighbor reports inside beacons to optimize the scanning process. In particular, when a neighbor report contains a known SSID, then the client can immediately scan this channel to see if that network is also within range of itself. If so, the client can immediately connect to this network, which greatly reduces the average time-to-connect. In case the network is not with in range of the client, we continue with the normal scanning process.

6.2 Experiments and Results

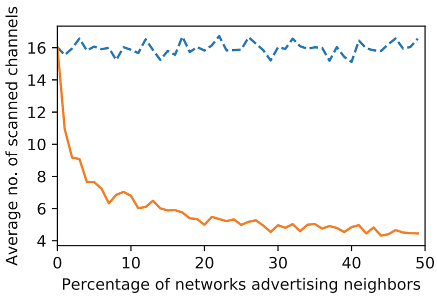
To estimate the benefits of including neighboring networks in beacon frames, we simulated this strategy and determined how much it reduced the average scanning time of a client. This allows us to determine how many APs must support this new feature for it to have a real benefit in practice. Note that in practice APs may not often get updated, meaning it is important that our new scanning strategy works well even if few APs advertise neighboring networks.

To determine the average scanning performance, we determine the number of channels that must be scanned before a known network is detected. In order to have realistic estimations of the number of nearby networks in our simulation, we use real-world Wi-Fi network locations from OpenWifi[3]. This is an open source database of Wi-Fi networks that is normally used for geolocation purposes. Based on this database, we perform the following steps in our simulation:

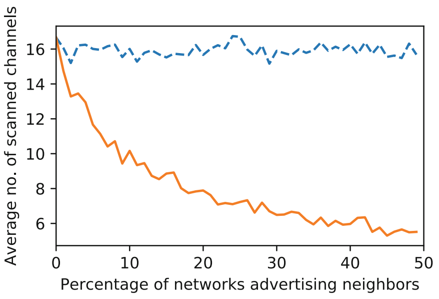
1. We assign a random operating channel to every network. Here we consider a total of 11 possible channels in the 2.4 GHz range, and 20 channels in the 5 GHz range.
2. A given percentage of networks is assumed to implement our proposal and advertise neighboring networks in their beacon frames.
3. We then randomly pick a position in a city, and determine all networks that are within 100 m of this position. These networks are considered to be within range of the client. We further assume a known network is nearby, since otherwise the scanning time will always equal the duration of a full network scan.
4. Additionally, we assume a second known network is out of range of the client, but within range of one or more nearby APs. This means this network may appear in the neighbor lists of APs, but will not be detected when the client searches for this network on the advertised channel.

Figure 6 shows the resulting number of scanned channels needed to find a known nearby network, in function of how many APs include neighboring networks in their beacon frames. What is surprisingly is that even if only a minor number of APs advertise nearby networks, this already results in a major reduction in scanning time. For instance, in San Francisco our results show that even if

[3] https://openwifi.su/download.php?lang=en.



(a) San Francisco, California.



(b) Boulder, Colorado.

Fig. 6. Average number of scanned channels before discovering a known network, when using normal scanning (dashed line) or when using neighbor advertising (solid line). Figure (a) contains the results for a densely populated area in San Francisco, while Figure (b) shows the results of a more sparse area in Boulder.

only 5% of APs advertise nearby network, then this already reduces the scanning time by more than half. To study the impact is less dense cities, we performed the same simulation in Boulder, and again found that if a minor number of APs advertise nearby networks, this already results in a major decrease of the scanning time. In particular, when only 10% of networks advertise its neighbors, then the average scanning time is almost halved.

Based on these experiments, we conclude that letting APs advertise nearby networks will result in a major decrease of the average scanning time. As a result, even when using passive scanning, known networks will be quickly found.

7 Related Work

Several researchers investigated the discovery process of Wi-Fi platforms and clients [1,6,9,10,12,18]. They conclude that most employ active scanning [10]. Unfortunately, capturing probe requests is trivial, and they provide a significant amount of information, ranging from family names, visited locations, travel routes, and so on [2,3,21].

To prevent tracking, modern devices use MAC address randomization when sending probe requests [17,23]. However, this defense can be (partly) bypassed. First, early implementations of address randomization did not reset the sequence number of probes between different individual scans [10]. Random MAC addresses can then be linked together by their incremental sequence numbers. Another method to link randomized probe requests is by relying on the included Information Elements (IEs). In particular, researchers found that the set of included IEs, their order, and their values, form a reliable fingerprint [23]. They suggest to avoid this by only including essential IEs. Finally, the timing between different network scans can also be used to track devices [18]. More precisely, the Inter-Frame Arrival Time (IFAT) between frames sent on the same channel within an individual scan forms a fingerprint of the device. This can be mitigated by sending probe requests with random delays [18]. In contrast, we look at the time between probe requests sent of *different* channels. Additionally, none of these works focus on improving passive scanning such that it becomes a viable alternative to active scanning.

Other works optimize active scanning by modifying the dwell time [1,6]. Their results indicate that in an environment with many APs, increasing the dwell time leads to a higher AP discovery rate [1]. However, increasing the dwell time past 100 ms did not yield more APs when using active scanning. Kim et al. propose to use the current location of a device to reduce privacy leaks during active scanning [15]. Here a device remembers the location of known networks, and only sends probe requests to networks that are likely nearby. Again, none of these works investigate passive scanning.

8 Conclusion

In this paper we argued that improving active scanning, such that it is more privacy-friendly, is not the best way forward. Instead, we believe a better approach is to improve the speed of passive scanning. We hope this makes vendors more incentivized to use passive scanning as the new default.

In particular, we proposed incremental and priority scanning. We implemented these modifications on Android, and evaluated their performance using several metrics. These experiments indicate that passive scanning can match and even surpass the speed of active scanning. For instance, when using static or dynamic priority scanning, the average time-to-connect is lower than default active scanning. This makes passive scanning a practical alternative to active scanning, improving privacy, and freeing up airtime.

Acknowledgments. Gunes Acar and Mathy Vanhoef hold a Postdoctoral fellowship from the Research Foundation Flanders (FWO). This work is partially supported by the Research Fund KU Leuven and by the Center for Cyber Security at New York University Abu Dhabi (NYUAD).

References

1. Arcia-Moret, A., Molina, L., Montavont, N., Castignani, G., Blanc, A.: Access point discovery in 802.11 networks. In: IFIP WD (2014)
2. Barbera, M.V., Epasto, A., Mei, A., Perta, V.C., Stefa, J.: Signals from the crowd: uncovering social relationships through smartphone probes. In: IMC (2013)
3. Bonne, B., Barzan, A., Quax, P., Lamotte, W.: WiFiPi: involuntary tracking of visitors at mass events. In: WoWMoM Workshop (2013)
4. Brik, V., Banerjee, S., Gruteser, M., Oh, S.: Wireless device identification with radiometric signatures. In: MobiCom (2008)
5. Campbell-Dollaghan, K.: Brave new garbage: London's trash cans track you using your smartphone (2013)

6. Castignani, G., Arcia, A., Montavont, N.: A study of the discovery process in 802.11 networks. ACM Mob. Comput. Commun. Rev. **15**(1), 25–36 (2011)
7. Cisco: Dynamic channel assignment (DCA). www.cisco.com/c/en/us/td/docs/wireless/controller/technotes/8-1/mobility_express/b_RRM_White_Paper/b_RRM_White_Paper_chapter_0100.pdf. Accessed 1 Aug 2019
8. Cisco Systems: Channel deployment issues for 2.4-GHz 802.11 WLANs (2004). xenguard.com/library/wifi/wifi-channels.pdf. Accessed 16 July 2018
9. Franklin, J., McCoy, D., Tabriz, P., Neagoe, V., Randwyk, J.V., Sicker, D.: Passive data link layer 802.11 wireless device driver fingerprinting. In: USENIX Sec (2006)
10. Freudiger, J.: How talkative is your mobile device? An experimental study of Wi-Fi probe requests. In: WiSec (2015)
11. Greenstein, B., McCoy, D., Pang, J., Kohno, T., Seshan, S., Wetherall, D.: Improving wireless privacy with an identifier-free link layer protocol. In: MobiSys (2008)
12. Gupta, V., Beyah, R., Corbett, C.: A characterization of wireless NIC active scanning algorithms. In: WCNC (2007)
13. IEEE Std 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Spec (2016)
14. Khoury, P.: Multiple BSSID support. In: IEEE 802.11-16/0586r1 (2016)
15. Kim, Y.S., Tian, Y., Nguyen, L.T., Tague, P.: LAPWiN: location-aided probing for protecting user privacy in Wi-Fi networks. In: CNS (2014)
16. Lindqvist, J., Aura, T., Danezis, G., Koponen, T., Myllyniemi, A., Mäki, J., Roe, M.: Privacy-preserving 802.11 access-point discovery. In: WiSec (2009)
17. Martin, J., et al.: A study of MAC address randomization in mobile devices and when it fails. PETS **2017**(4), 365–383 (2017)
18. Matte, C., Cunche, M., Franck, R., Vanhoef, M.: Defeating MAC address randomization through timing attacks. In: WiSec, July 2016
19. Microsoft: Non-broadcast wireless SSIDs: why hidden wireless networks are a bad idea (2008). blogs.technet.microsoft.com. Accessed 16 July 2018
20. Nicholson, A.J., Noble, B.D.: Breadcrumbs: Forecasting mobile connectivity. In: MobiCom (2008)
21. Pang, J., Greenstein, B., Gummadi, R., Seshan, S., Wetherall, D.: 802.11 user fingerprinting. In: MobiCom (2007)
22. Peddemors, A., Eertink, H., Niemegeers, I.: Predicting mobility events on personal devices. Pervasive Mob. Comput. **6**(4), 401–423 (2010)
23. Vanhoef, M., Matte, C., Cunche, M., Cardoso, L.S., Piessens, F.: Why MAC address randomization is not enough: an analysis of Wi-Fi network discovery mechanisms. In: Asia CCS (2016)
24. ZyXel: Dynamic channel selection (DCS). https://www.zyxel.com/uploads/Dynamic_Channel_Selection_4.20.pdf. Accessed 1 Aug 2019

Cronus: Everlasting Privacy with Audit and Cast

Thomas Haines(✉)

Norwegian University of Science and Technology, Trondheim, Norway
thomas.haines@ntnu.no

Abstract. We present a new online voting scheme with everlasting privacy and cast-as-intended verifiability. We follow the so called "audit-and-cast" paradigm where the voter audits the ballot before casting it. To mitigate the ability of this information to harm the voter's privacy, we provide measures for avoiding coercion by allowing **any** party to create fake proofs for the content of **any** vote. We propose an efficient implementation and formally verify its security properties.

1 Introduction

This work focuses on simultaneously achieving two properties, everlasting privacy and cast-as-intended verification—along with end-to-end verifiability—in an efficient manner. Everlasting privacy is achieved because the public information in the scheme is unconditionally hiding. Further, our scheme provides timely and accountable cast-as-intended verification which the voter confirms before the ballot is accepted; this results in a simple vote ceremony which is dispute free. This prevents the issue found in many other schemes, where disgruntled voters can cast aspersions which cannot be resolved. While we do not claim that our scheme provides coercion resistance, we do provide to all parties the ability to create fake proofs of all votes. This means interested parties can run counter-coercion strategies on behalf of voters.

2 Related Work

Electronic voting schemes have been the subject of much cryptographic study since the 1980s [3,7–9,11,12,24]. During this extensive study of verifiable electronic voting schemes many subtle issues have been detected. One such issue is receipt-freeness [4] and another is cast-as-intended verifiability. Unfortunately the former significantly complicates the latter. Prominent verifiable electronic voting schemes for remote voting like Helios [2] use Benaloh challenges [32] to achieve cast-as-intended verification without generating receipts, but this

The author acknowledges support from the Luxembourg National Research Fund (FNR) and the Research Council of Norway for the joint project SURCVS. Part of this work was completed while the author was working at Polyas GmbH.

A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 53–68, 2019.
https://doi.org/10.1007/978-3-030-35055-0_4

mechanism—as currently implemented—is (often) unusable [21]. Compounding the usability issues with some cast-as-intended mechanisms are the further issues of accountability and dispute management. We desire that electronic voting schemes provide strong evidence of misconduct, but the definitions of cast-as-intended verification require only that a voter can detect misconduct. While this issue is dealt with in some papers [10,30], in many deployed examples and papers [19,29,31] the definitions do not imply strong evidence. This lack of strong evidence means that if honest voters make mistakes in verification, or disgruntled voters desire to cast aspersions on the security of the election, there is no way to tell the difference between these scenarios and an attack. To provide strong evidence the definition needs to be strengthened to something closer to accountability [22].

Many voting protocols have been proposed with everlasting privacy and others have configurations which achieve this property. Everlasting privacy was proposed as an extension to the Helios scheme by Demirel et al. [14], extending the Split-ballot voting scheme from Moran and Naor [25]. This kind of extension reduces privacy attacks on the system (from an external adversary) to information theoretic security rather than computational. Hence, no future breakthrough in computation power, mathematics, or large scale quantum computers will put the voters' privacy at risk to the public. While there are schemes [35] which provide information theoretic privacy these are impractical for most/all real elections. Demirel et al.'s scheme and many others, including our work here, have at least one authority against which privacy holds only computationally. Another set of schemes use some form of anonymous signature (ring, group, linkable) and an anonymous channel [23] which achieves everlasting privacy cleanly, but the existence of such a channel is problematic to realise [34].

Our work is similar to both Selene [29] and Guasch et al.'s work on "How to Challenge and Cast Your e-Vote" [17] in that we make use of a voter controlled trapdoor which allows the voter to be given a proof about which they can then equivocate to possible coercers. Our work provides stronger cast-as-intended verifiability than either of the above by avoiding dispute issues in the cast-as-intended mechanism. We achieve this by getting the voter to confirm that they are happy with the cast-as-intended check before submitting. Selene has a significantly different user experience where the voter checks that their vote was included after the election is complete, whereas Guasch et al.'s work has a similar voter experience to ours. In addition, while there seems to be no insurmountable barrier to updating either of the above schemes to have everlasting privacy, neither scheme currently has this property.

3 Building Blocks

3.1 Definitions

We begin by stating the standard definitions.

Definition 1. *Encryption Scheme: An encryption scheme Π_ϵ consists of a triple of efficient algorithms $(Gen_\epsilon, Enc_\epsilon, Dec_\epsilon)$ such that:*

$Gen_\epsilon(1^n)$**:** *Given the security parameter* 1^n*, output a pair of public and secret keys* (pk, sk)*. The public key specifies a message space* M_ϵ*.*

$Enc_\epsilon(pk, m)$**:** *Given a public key and a message* m*, return* $\perp$ *if* $m \notin M_\epsilon$*, otherwise output a ciphertext* c*. The explicit notation is* $Enc_\epsilon(pk, m; r)$ *for the randomness used in the encryption.*

$Dec_\epsilon(sk, c)$**:** *Using the private key* sk*, output a decryption* d *of the ciphertext* c*.*

We require an encryption system to be correct and at least preserve indistinguishability of messages under chosen plaintext attacks.

Definition 2. *Trapdoor Commitment Scheme: A trapdoor commitment scheme* Π_c *is a tuple of efficient algorithms* $(Gen_c, GenTK_c, Com_c, Verify_c, Fake_c)$ *such that:*

$Gen_c(1^n)$**:** *Given the security parameter* 1^n*, output a public commitment key* ck *and a proof that* ck *was generated without a known trapdoor. The commitment key specifies a message space* M_c*.*

$GenTK_c(1^n)$**:** *Given the security parameter* 1^n*, output a public commitment key* ck *and trapdoor key* tk*. The commitment key specifies a message space* M_c*.*

$Com_c(ck, m)$**:** *Given a commitment key and a message* m*, return* $\perp$ *if* $m \notin M_c$*, otherwise output a pair* (c, d) *made of a commitment and an opening. The explicit notation is* $Enc_\epsilon(ck, m; r)$ *for the randomness used in the commitment.*

$Verify_c(ck, c, d, m)$**:** *Given a commitment key, commitment, opening and message, return either 1 or 0 if these inputs are consistent or not.*

$Fake_c(ck, tk, c, d, m, m')$**:** *Given a commitment key, trapdoor key, commitment, opening and messages* m *and* m' *returns a new opening* d'*.*

Clearly the commitment scheme should be correct and it should not be possible to open the same commitment to two different messages without knowledge of the trapdoor key. In addition, our scheme requires that the commitments be statistically hiding.

Definition 3. *Signature Scheme: A signature scheme* Π_s *is a triple of efficient algorithms* $(Gen_s, Sign_s, Verify_s)$ *such that:*

$Gen_s(1^n)$**:** *Given a security parameter* 1^n*, output a pair of public and secret keys* (pk, sk)*. The public key specifies a message space* M_s*.*

$Sign_s(sk, m)$**:** *Given a secret key and a message* m*, return* $\perp$ *if* $m \notin M_s$*, otherwise output a signature* s*.*

$Verify_s(pk, m, s)$**:** *Return either 1 if the signature verifies for the public key or 0 if not.*

A signature scheme should be correct and prevent even existential forgeries.

3.2 Specifics

We take as our primary building blocks the Pedersen commitment scheme [28] and Paillier encryption [27]. These have been very commonly used in e-voting with everlasting privacy, going back to Moran's seminal work [25], and were also used in [14] and [13]. In essence, one commits to the vote in a Pedersen commitment and then encrypts the opening under Paillier encryption. The unconditionally hiding Pedersen commitments can ensure the integrity of the election on the public board without revealing any information about the vote. In addition, we make use of verifiable mixnets and sigma protocols.

We note that Paillier has somewhat of a bad reputation in the e-voting community for complicating secure threshold key generation. While this was certainly a significant concern when Paillier was first proposed, subsequent research has changed the situation [26]; of particular note in addressing this concern is Hazay et al.'s [20] recent work. At present, the remaining reason not to prefer Paillier encryption is computational costs which we would argue are insufficiently problematic to justify the procedural issues incurred by most alternatives.

We note in passing that it is possible to adapt the very elegant PPATC scheme of Cuvelier et al. [13] for use with our scheme as a replacement for Paillier encryption and Pedersen commitment. The resulting scheme proceeds identically to how we describe, but instead of Pedersen commitments, Abe et al.'s [1] commitment scheme is used—once with known trapdoors and once without. This works because while PPATC is not additively homomorphic, an analogous sigma protocol to the "Sigma protocol for consistent commitments"—which we will define below—is available. We detail this sigma protocol in Appendix A.

Paillier encryption consists of a triple of efficient algorithms ($Gen_{Pa}, Enc_{Pa}, Dec_{Pa}$) such that:

$Gen_{Pa}(1^n)$**:** Choose two n-bit primes, p and q, and compute the public modulus $N := pq$ for the Paillier encryption. The corresponding secret key $\lambda := \lambda(N)$ is the number $lcm(\phi(p), \phi(q)) = \phi(N)/2$. Output (N, λ).

$Enc_{Pa}(N, m)$**:** For $m \in \mathbb{Z}_N$ choose $r \in_R \mathbb{Z}_N^*$ and return $= (N+1)^m r^N \bmod N^2$.

$Dec_{Pa}(\lambda, c)$**:** For $c \in \mathbb{Z}_{N^2}$ return $([c^\lambda \bmod N^2] - 1)/N)[\lambda^{-1} * x \bmod N] \in \mathbb{Z}_N$.

Pedersen's commitment scheme consists of a tuple of efficient algorithms (Gen_{Pe}, $GenTK_{PE}$ $Com_{Pe}, Verify_{Pe}, Fake_{Pe}$) such that:

$Gen_{Pe}(1^n)$**:** Choose a group G of order o, $2^{n-1} < o < 2^n$, in which the discrete log problem is hard. Select two generators g and h in a verifiably random way such that no non-trivial information about the relationship between is revealed, for instance as described in [16]; return $ck := (G, o, g, h)$. We also allow G to be given explicitly in which case the notation is $Gen_{Pe}(G)$.

$GenTK_{PE}(1^n)$**:** Choose a group G of order o approximately equal to 1^n in which the discrete log problem is hard. Select a random generator g and random element of $\mathbb{Z}_o$ x and set $h = g^x$; return $ck := (G, o, g, h)$ and $tk := x$. We also allow G to be given explicitly in which case the notation is $Gen_{Pe}(G)$.

$Com_{Pe}(ck, m)$**:** Given a commitment key and message $m \in \mathbb{Z}_o$ choose $r \in_R \mathbb{Z}_o$ and return $(g^r h^m, r)$.

$Verify_{Pe}(ck, c, d, m)$**:** Given a commitment key, commitment c, opening d and message m return 1 if $c = g^d h^m$ else 0.

$Fake_{Pe}(ck, tk, c, d, m, m')$**:** Recall that $tk = x$ where $h = g^x$, return $(r' := r + xm - xm')$.

In our scheme we use Pedersen commitments in two separate places (with different commitment keys). First, we use them to achieve everlasting privacy while preserving integrity. In this use, the ability to implant a trapdoor is undesirable, so the setup should produce two generators which are verifiably drawn uniformly and independently at random. They should be drawn in such a way that no more information about the relationship between them is revealed than can be easily computed given only the two elements. However, we also use the Pedersen commitments to allow the voters to verify that their ballot is cast as intended but later equivocate about the proof, in which case we make deliberate use of the trapdoors. **We emphasise that since these two different places have two different sets of keys no contradiction occurs by having one trapdoored and the other not.**

We can exploit the homomorphic properties of the above primitives to create efficient Zero Knowledge Proofs (ZKP) of correct encryption. Specifically given a Pedersen commitment of the vote c, Paillier ciphertexts opening the commitment c_1 and c_2, and an additional verification Pedersen commitment c_v, we can prove that the ciphertexts do encrypt an opening to the commitment and that the additional verification commitment refers to the same vote. The sigma protocol for correct encryption first appeared in [13].

Sigma protocol for correct encryption. Given a $ck, pk, (c, c_1, c_2)$ we show that we know (v, r, r_1, r_2) such that $c = Com_{Pe}(ck, v; r)$, $c_1 = Enc_{Pa}(N, r; r_1)$, and $c_2 = Enc_{Pa}(N, v; r_2)$.

1. Prover chooses (v', r', r_1', r_2') at random and computes $c' = Com_{Pe}(ck, v'; r')$, $c_1' = Enc_{Pa}(pk, r'; r_1')$, and $c_2' = Enc_{Pa}(pk, v'; r_2')$ and returns (c', c_1', c_2').
2. Verifier sends a challenge e chosen at random in $\mathbb{Z}_N$.
3. Prover computes $t_1 := v' + ev$, $t_2 := r' + er$, $t_3 := r_1' {r_1'}^e$, and $t_4 := r_2' {r_2'}^e$ and sends these to the verifier.
4. The verifier accepts if $c'c^e = Com_{Pe}(ck, t_1; t_2)$ and $c_1' c_1^e = Enc_{Pa}(pk, t_2; t_3)$ and $c_2' c_2^e = Enc_{Pa}(pk, t_1; t_4)$.

Sigma protocol for consistent commitments. Given a $ck, ck_v, (c, c_v)$ we show that we know (v, r, r_v) such that $c = Com_{Pe}(ck, v; r)$, and $c_v = Com_{Pe}(ck_v, v; r_v)$.

1. Prover chooses (v', r', r_v') at random and computes $c' = Com_{Pe}(ck, v'; r')$, and $c_v' = Com_{Pe}(ck_v, v'; r_v')$ and returns (c', c_v').
2. Verifier sends a challenge e chosen at random in $\mathbb{Z}_N$.
3. Prover computes $t_1 := v' + ev$, $t_2 := r' + er$, and $t_3 := r_v' + er_v$ and sends these to the verifier.
4. The verifier accepts if $c'c^e = Com_{Pe}(ck, t_1; t_2)$ and $c_v' c_v^e = Com_{Pe}(ck_v, t_1; t_3)$.

We can make the sigma protocol for correct encryption non-interactive by applying the Fiat-Shamir transform, and we will in future refer to **CEProve** and **CEVerify** for the non-interactive prover and verifier functions respectively. We do the same for the consistent commitment sigma protocol and refer to **CCProve** and **CCVerify**.

The Sigma protocol for correct encryption above can be modified to prove correct re-encryption of a tuple of a Pedersen commitment and two Paillier ciphertexts. Given the sigma protocols for correct re-encryption we can apply Wikström's general result from [33] to construct a mixnet. An optimised variant of Wikström's mixnet for shuffling Pedersen and Paillier together recently appeared in [18].

We will refer to the mixnet for the Pedersen commitment and Paillier ciphertexts together as **Mix** and its verification algorithm as **MixVerify**; similarly we denote the mixnet for commitments alone as **Mix'** and its verification algorithm as **MixVerify'**. We denote **MixSimulate** and **MixSimulate'** the simulators for the mixnets which are obtained by reprogramming the random oracle. It is a trivial equivalence for the authorities to check that the commitments are shuffled according to the same permutation and updated by the same randomness factors on both boards. We note that in this case, the structure of Wikström's proof also works for a significantly optimised variant of the mixnet from the general result, as is also true in the case of ElGamal for instance; we omit the details.

4 Cronus E-Voting Scheme

We now describe the scheme which uses two bulletin boards BB and sBB; the first is a public board and the second can only be seen by the authorities. Since BB is public information all algorithms are assumed to have access to all its contents. For simplicity we describe it with a single key holder although threshold key generation is available:

Setup(1^n) runs $Gen_{Pa}(1^n)$ and receives (N, λ), chooses k such that $kN+1$ is prime, denoting the subgroup of order N in $\mathbb{Z}_{kN+1}$ as G and runs $Gen_{Pe}(G)$ and receives $(g,\ h)$, and sets the election public key $pk = (N, G, g, h)$ and $sk = (\lambda)$. Then it generates the empty list of credentials **ID** and posts pk and **ID** BB.

Register$(1^n$, id) runs $GenTK_{Pe}(G)$ and $Gen_s(1^n)$ and sets $pk_{id} = (ck, pk_s)$ and $sk_{id} = (tk, sk_s)$, and posts (id, pk_{id}) to BB unless id already appears on the board in which case it aborts.

CreateVote(v, id) retrieves $(pk = (N, G, g, h), pk_{id} = (ck, pk_s))$ from BB. Then, it runs $Com_{Pe}((g, h), v)$ and receives (c, r); it then chooses r_1 and r_2 at random in $\mathbb{Z}_N^*$, then runs $Enc_{Pa}(N, r; r_1)$ and $Enc_{Pa}(N, v; r_2)$ and receives c_1 and c_2. It then runs $Com_{Pe}(ck, v)$ and receives (c_v, r_v). The voting device calls **CEProve**$(c, c_1, c_2)(v, r, r_1, r_2)$ and **CCProve**$(c, c_v)(v, r, r_v)$ and receives π_{CE} and π_{CC}. The ballot b is then $(c, c_1, c_2, c_v, \pi_{CE}, \pi_{CC})$. The device also outputs (v, r_v) which will be used to audit the ballot.

CastBallot(b, sk_{id}, id) runs $Sign_{sk_{id}}(b)$ and receives ζ. The authenticated ballot is then $b_a = (id, b, \zeta)$.

ProcessBallot(b_a) parses b_a as (id, b, ζ) and b as $(c, c_1, c_2, c_v, \pi_{CE}, \pi_{CC})$. It checks that $Verify_s(pk_{id}, b, \zeta) = 1$ and that **CEVerify**$(c, c_1, c_2, \pi_{CE}) = 1$ and that **CCVerify**$(c, c_v, \pi_{CC}) = 1$. If tk_{id} is not present BB then posts (id, c, c_v, π_{CC}) to the public bulletin BB and (id, c_1, c_2, π_{CE}) to the private bulletin board sBB.

AuditVote(id, v, r_v)**.** On receiving (id, v, r_v), the audit device accesses the bulletin board to retrieve (c, c_v, π_{CC}). It checks that **CEVerify**$(c, c_v, \pi_{CC}) = 1$ and that $Verify_{Pe}(ck_{id}, c_v, v, r_v) = 1$. If these checks pass it returns 1, otherwise 0.

ConfirmBallot$(id, sk_{id}, (v, r_v))$ parses sk_{id} as (tk, sk_s) and retrieves pk_{id} parsing it as (ck, pk_s). It then checks that tk is valid for ck and that $Verify_{Pe}(ck, c_v, v, r_v) = 1$, if so it chooses v' at random from the set of valid votes and posts (tk, v', r'_v) to the BB where $r'_v = Fake_{Pe}(ck, tk, c_v, r_v, v, v')$, otherwise it returns 0.

Tally$(sk,\ sBB)$

Filter. First the authorities filter out the ballots which were not confirmed.

Parallel shuffle. They then take the set of commitments on the public board and the set of commitments taken with the ciphertexts on the private board. They then re-randomise and shuffle these sets using **Mix'** and **Mix** respectively, checking that the commitments on the two boards match at each step. We denote by Φ_1 and Φ_2 the Non-Interactive Zero Knowledge Proof (NIZKP) proofs of correct mixing for the public and private board respectively. The output set of commitments along with intermediary values and Φ_1 are posted to the public board BB and the output set of ciphertexts and Φ_2 are posted to the secret board sBB.

Decryption. The authorities then jointly decrypt the set of ciphertexts of the form (c'_1, c'_2) to recover (r', v) which are then posted to the public board.

VerifyTally(pk) to verify that a set of votes $(v_1, ..., v_n)$ are the correct result, an auditor retrieves the input commitments, output commitments, intermediary values, openings and Φ_i for the public board. It then runs **Mix'** and checks that $Verify_{Pe}(ck, c', v, r') = 1$ for all commitments c' and openings r'. If all checks pass it returns 1, else 0.

4.1 Election Flow

The election protocol has the following participants: *Election Authorities*, *Registrars*, and *Voters* whom we assume have access to a *Voting device* and *Audit device.*

Setup Phase. Before an election, the set of election authorities set up the public parameters as defined by **Setup**, which they publish to the public bulletin board BB—and also define the voting options and the tally function. The voters are registered by the registrars, using the **Register** function, and the voters pk_{id} are posted to the BB.

Vote Casting Phase. During the vote casting stage, voters access their voting devices and cast their ballots using **CreateVote** and **CastBallot**, and receive v, r_v. They then provide v, r_v to the audit device which checks the ballot using **AuditBallot**. If the ballot audit is successful, the voter then confirms their ballot using **Confirm Ballot**. The voter should check that both their voting device and audit device agree that the ballot has been confirmed.

Tallying Phase. After voting is over, the authorities run **Tally** which runs the mixnets and tallies the votes.

Audit Phase. After the election, any party can check the signatures on the submitted ballots to see that ballots were collected as cast and no ballots from ineligible voters are present. They can also run **VerifyTally** to check that the ballots were counted as collected.

The voter's experience in the most straightforward instantiation of the scheme is of authenticating and voting on one device, which then displays a QR code. The voter then scans this with a second device which confirms that the encrypted ballot encodes the voter's choice. The voter then confirms their ballot.

5 Security Definitions and Analysis

In defining Ballot Privacy, we follow the Ballot PRIVacy (BPRIV) definitions of Bernhard et al. [5] which we have slightly modified, while our cast-as-intended definition is similar to Escala et al. [15,17]. We also briefly discuss why everlasting privacy holds and why the encryption is Non-Malleable and indistinguishable under a Chosen Plaintext Attack (NM-CPA).

5.1 Ballot Privacy

The intuition for the formal definition of privacy is that privacy should hold even against insiders up to and including any subset of the authorities, less than the threshold that can recover the key—that threshold can trivially break privacy by decrypting the ciphertexts next to the voter id. Formally we define ballot privacy by the adversary advantage in the following experiment. Note that in our privacy definitions we prepend the bulletin board on which the algorithm is running to its list of inputs when necessary for clarity.

```
$\mathbf{Exp}^{bpriv,\beta}_{\mathcal{B},\mathcal{V}}(n)$
    $(pk, sk) \leftarrow \mathbf{Setup}(1^n)$
$\mathcal{O}voteLR(id, v_0, v_1)$
    $sk_{id} = \mathbf{Register}(1^n, id)$
    Let $(b_0, (v_0, r_0)) = \mathbf{CastBallot}(\mathbf{VoteCreate}(v_0, id), sk_{id}, id)$ and
        $(b_1, (v_1, r_1)) = \mathbf{CastBallot}(\mathbf{VoteCreate}(v_1, id), sk_{id}, id)$
    If $\mathbf{ProcessBallot}(b_\beta) = 0$ return 0.
    If $\mathbf{ConfirmBallot}(id, sk_{id}, (v_\beta, r_\beta)) = 0$ return 0.
$\mathcal{O}cast(id, b, (v, r))$
    $sk_{id} = \mathbf{Register}(1^n, id)$
    If $\mathbf{ProcessBallot}(b) = 0$ return 0.
    If $\mathbf{ConfirmBallot}(id, sk_{id}, (v, r)) = 0$ return 0.
$\mathcal{O}board()$
    return $(BB_\beta, sBB_\beta)$
$\mathcal{O}tally()$ for $\beta = 0$
    $(r, \phi) \leftarrow \mathbf{Tally}(BB_0, sBB_0, sk)$
$\mathcal{O}tally()$ for $\beta = 1$
    $(r, \phi) \leftarrow \mathbf{Tally}(BB_0, sBB_0, sk)$
    $\phi' \leftarrow \mathbf{SimProof}(BB_1, r)$
    $return(r, \phi')$
```

Definition 4. *BPRIV Consider a voting scheme $\mathcal{V} = ($**Setup**, **Register**, **CreateVote**, **CastBallot**, **ProcessBallot**, **AuditVote**, **ConfirmBallot**, **Tally**, **VerifyTally**$)$ for a set I of voter identities for a result function p. We say the scheme has ballot privacy if there exists an algorithm **SimProof** such that no efficient adversary can distinguish between games $\boldsymbol{Exp}^{bpriv,0}_{\mathcal{B},\mathcal{V}}(\lambda)$ and $\boldsymbol{Exp}^{bpriv,1}_{\mathcal{B},\mathcal{V}}(\lambda)$ defined by the oracles above, that is for any efficient algorithm $\mathcal{A}$*

$$|Pr[\boldsymbol{Exp}^{bpriv,0}_{\mathcal{B},\mathcal{V}}(\lambda) = 1] - Pr[\boldsymbol{Exp}^{bpriv,1}_{\mathcal{B},\mathcal{V}}(\lambda)]|$$

is negligible in λ.

Due to similarities between between Helios and our scheme the proof of ballot privacy is similar. However, there are significant differences; we have a public and a confidential bulletin board and a two-stage ballot casting process. For the purpose of proving ballot privacy we assume casting and confirming a ballot is an atomic process, since the adversary has less information to attack privacy when his process aborts rather then completes this does not enable any attacks. We define **BB** as the union of BB and sBB so that it contains entries of the form $(id, b = (c, c_1, c_2, c_v))$.

Recall the observation that since the visible bulletin board is built through the $\mathcal{O}voteLR(id, v_0, v_1)$ and $\mathcal{O}cast(id, b, (v, r))$ queries, our BPRIV reduction can associate, to any entry (id_i, b_i) in the visible bulletin board, a tuple $(id_i, b_i^0, b_i^1, v_i^0, v_i^1)$.

[**Game** G_{-1}] Let G_{-1} be the BRPIV game corresponding to Experiment $\mathbf{Exp}^{bpriv,0}_{\mathcal{B},\mathcal{V}^{Cronus}}$. The BPRIV adversary $\mathcal{A}$ sees the ballot box $\mathbf{BB}_0$ and an oracle $\mathcal{O}tally()$ faithfully answered.

[**Game** G_0] Let G_0 be the same as Game G_{-1} except the tallying proof ϕ' is produced by **MixSimulate** and **MixSimulate'** by reprogramming the Random Oracle G. Due to the zero-knowledge property of the NIZKP proof associated with the mixnet, the distinguishing probability is negligibly close between the two games. From now on the proof is always simulated.

[**Game** $G_{0,i}$] is obtained from Game $G_{0,i-1}$ by taking two possible actions depending on the contents of the tuple $(id_i, b_i^0, b_i^1, v_i^0, v_i^1)$: if $b_i^0 = b_i^1$ do nothing, else $b_i^0 \neq b_i^1$ replace the i-th entry (id_i, b_i^0) in BB_0 with (id_i, b_i^1). By the NM-CPA property of our construction, the distinguishing probability of the adversary is negligibly close to $G_{0,i-1}$.

[Game G_1**]** Let G_1 be Game $G_{0,n}$. The view of the adversary in Game G_1 corresponds to the view of the BPRIC adversary with $\beta = 1$. Cronus is thus BPRIV private.

5.2 Cast-as-Intended Verifiability

The challenger $\mathcal{C}$ calls **Setup**(1^λ) and provides (N, G, g, h) to the adversary. For a list of voters ID and Bulletin boards BB, sBB, it provides the following oracles.
$\mathcal{O}registerHonest(id)$
 $\mathcal{A}$ provides $id \notin ID$. The challenger $\mathcal{C}$ calls **Register**$(1^\lambda, id)$ adding (id, pk_{id}) to ID.
$\mathcal{O}registerCorrrupt(id, pk_{id})$
 $\mathcal{A}$ provides $id \notin ID$. The challenger $\mathcal{C}$ adds (id, pk_{id}) to ID.
$\mathcal{O}cast(id, b)$
 $\mathcal{C}$ returns **CastBallot**(b, sk_{id}, id).
$\mathcal{O}process(b_a)$
 $\mathcal{C}$ returns **ProcessBallot**(b_a).
$\mathcal{O}confirmHonest(id, v, r_v)$
 If **AuditVote**(id, v, r_v) returns 1 then return sk_{id} and run **ConfirmBallot**$(id, sk_{id}, (v, r_v))$.
$\mathcal{O}confirmCorrupt(id, b_a)$
 return sk_{id}.

Definition 5. *Cast-as-Intended Verification: Consider a voting scheme $\mathcal{V}$ = (**Setup**, **Register**, **CreateVote**, **CastBallot**, **ProcessBallot**, **AuditVote**, **ConfirmBallot**, **Tally**, **VerifyTally**) for a set ID of voter identities and a result function p. We say the scheme has cast-as-intended verifiability if there exists no efficient adversary that can win the following game with greater than negligible probability.*

The adversary wins if there exists a confirmed ballot $(id, c, c_v) \in BB$ and $(id, c_1, c_2) \in sBB$ such that the following conditions hold:

- $id \in ID$ and id was not the input to $\mathcal{O}registerCorrupt$.
- $Dec_{Pa}(sk, c_2) \neq v$ where v was the voting option submitted by the adversary to $\mathcal{O}confirmHonest$.

Provided that Cronus is instantiated with a secure signature scheme $(Gen_s, Sign_s, Verify_s)$, sound Pedersen commitment scheme $(Gen_{Pe}, GenTK_{Pe}, Com_{Pe}, Verify_{Pe}, Fake_{Pe})$, and sound NIZKP schemes (**CEProve**, **CEVerify**), (**CCProve**, **CCVerify**)—and one of the devices is honest—no such efficient adversary exists.

First note that all entries on the board, confirmed or otherwise, were placed there by the $\mathcal{O}process$ oracle. This means that the adversary has shown that it knows an opening v, r, r_1, r_2 such that $c = Com_{Pe}(ck, v; r)$, $c_1 = Enc_{Pa}(N, r; r_1)$ and $c_2 = Enc_{Pa}(N, v; r_2)$; and, that it knows an opening v', r', r_v such that $c = Com_{Pe}(ck, v'; r')$ and $c_v = Com_{Pe}(ck, v'; r_v)$. By the binding property of the Pedersen commitments $v = v'$ and $r = r'$.

Secondly, observe that all entries on the board not violating the first condition were confirmed there through the $\mathcal{O}confirmHonest$ oracle. This oracle checks that the adversary can open the commitment c_v to v before it confirms the ballots. Since the Pedersen commitments are binding, this v must be the only opening which the adversary knows; and, hence, the same v which it showed to be equal to contents of the ciphertext c_2 when the ballot was processed. Therefore, by the soundness of **CEProve** and **CCProve** and the binding property of the commitment scheme, the vote v submitted to $\mathcal{O}confirmHonest$ is the vote which c_2 decrypts to.

We have just shown that the first device is unable to submit a different vote without being detected, but what if the second device is corrupt? If both devices are corrupt then the voter has no integrity guarantees. However, it is also clear that if the audit device is corrupt but the voting device is honest the ballot is sent correctly and the voting device will honestly tell the voter when their ballot is confirmed. We note that if one device complains that the other has misbehaved, diagnosing which device is actually malicious is non-trivial.

5.3 Strong Consistency and Strong Correctness

We define the following extraction and verification algorithms consistent with Tally and ProcessBallot:

1. Extract$((c, c_1, c_2, c_v, \pi_{CE}, \pi_{CC}), sk)$ first verifies the proofs π_{CE} and π_{CC} and if either fails returns $\perp$. Otherwise decrypts c_2 and return the result.
2. ValidInd(b) checks that the ballot is valid and the proofs are correct. That is, given a ballot $b = (c, c_1, c_2, c_v, \pi_{CE}, \pi_{CC})$ it checks that both π_{CE} and $\pi_C C$ verify.

Definition 6. *Strong Consistency: A voting protocol has strong consistency if the following hold:*

- *For all pk from **Setup** for any voter v, for any voter identity id and pk$_i$d for **Register**(id), Extract(**CreateVote**(v,id)) = v.*
- $\boldsymbol{ProcessBallot}(b_a) = 1, \boldsymbol{AuditBallot}(id, v, r_v) = 1, \boldsymbol{ConfirmBallot}$ $(id, sk_id, (v, r_v)) = 1$ *implies that*$ValidInd(b) = 1.$
- *Where the adversary's chance of winning the following game is negligible,*
  - **Setup phase.** *The challenger runs **Setup**(1^n) to generate pk and sk, which it gives to $\mathcal{A}$.*
  - **Bulletin Board.** *$\mathcal{A}$ submits a bulletin board BB and sBB.*
  - **Counting phase.** *The challenger runs **Tally**(sk) and obtains the set of output votes r and tally proof ϕ_1.*

Output. *The adversary $\mathcal{A}$ wins if $r \neq Extract(BB, sk)$, where Extract is applied to each confirmed ballot on the bulletin board.*

The first property follows trivially from the correctness of Paillier encryption. The second property follows because **ProcessBallot** checks the same proofs as ValidInd. The last property follows trivially because the definition of extract and tally are the same up to mixing.

Definition 7. *Strong Correctness: A voting protocol has strong correctness if given $pk = \boldsymbol{Setup}(1^n)$, for any efficient algorithm $\mathcal{A}$, the following probability*

$$Pr[(id, v, BB) \leftarrow \mathcal{A}(pk); \boldsymbol{Register}(1^n, id); \boldsymbol{CreateVote}(v, id) = b, (v, r_v);$$
$$\boldsymbol{CastBallot}(b, sk_{id}, id) = b_a : \boldsymbol{ProcessBallot}(b_a) = 0 \vee \boldsymbol{AuditVote}(id, v, r_v) = 0$$
$$\vee\, \boldsymbol{ConfirmBallot}(id, sk_{id}, (v, r_v) = 0]$$

is negligible in λ, where id does not appear on BB.

By definition of **ProcessBallot**, **AuditVote** and **ConfirmBallot** the above definition implies that the following events must occur except with negligible probability. The signature produced by **CastBallot** must verify. The sigma protocol transcripts produced by **CreateVote** must verify. The tk_{id} must not already appear on the bulletin board. The commitment c_v must open to v, r_v. All of these events are implied by the correctness of the relevant primitives with the exception of tk_{id} not appearing on the bulletin board; for this, we require that id does not already appear on the BB as produced by the adversary.

5.4 Everlasting Privacy

Everlasting privacy is fairly straightforward. The public bulletin board BB contains the following information for each confirmed ballot at the end of the election $(id, pk_{id} = (ck, pk_s), c, c_v, \pi_{CC})$. The commitments c and c_v are statistically hiding and hence leak negligible information about the vote regardless of adversary's computational power. The proof π_{CC} is honest verifier zero knowledge and also leaks negligible information about the vote regardless of adversary computational power. Since no non-negligible information is leaked about the vote based on the public information, we conclude that the scheme has everlasting privacy.

5.5 Encryption

It is known that an Indistinguishable under Chosen Plaintext Attack (IND-CPA) secure cryptosystem plus a Simulation Sound Extractable Proof of Knowledge (SSE-PoK) is Non-Malleable under Chosen Plaintext Attack (NM-CPA) [6]. It also known that applying the strong Fiat-Shamir transform to sigma protocols yields an SSE-PoK. We therefore have that our construction is also NM-CPA secure provided that Paillier-Encryption is IND-CPA secure. It is also necessary to prevent simple duplication of ballots; this can be done by filtering duplicates or including the voter id in the input to the hash function challenge generator in the non-interactive version.

6 Practical Realisation

In practice the voter needs to have access to the keys in some way. One option is that the voter has a trusted authentication device which handles the signing and trapdoor for them; however, this reduces the practicality of the scheme. Another option is to provide the voter with a confirmation code and distribute the trapdoor among the set of tellers; the set of tellers would release the trapdoor when the voter confirms. Interestingly, this realisation does not affect the integrity of the scheme—unless the first device colludes with the tellers—but has a mild impact on privacy.

7 Conclusion

We have presented a straightforward and effective scheme with cast-as-intended verifiability, everlasting privacy, and universal verifiability. The scheme avoids many of the issues hitherto present in similar schemes through careful use of checks and zero-knowledge proofs. The construction as we present it relies on Pedersen commitments and Paillier encryption; however, it is equally possible to instantiate over elliptic curves of prime order with bilinear pairings.

A Sigma protocol for consistent Abe commitments

We present a sigma protocol which shows that the prover can open two of Abe et al.'s [1] commitments to the same message. Recall that Abe et al.'s commitments are defined over an elliptic curve coupled with a bilinear pairing; we denote the groups of the curve as $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. Given two generators for $\mathbb{G}_1$ denoted G_0, G_1 and a generator for $\mathbb{G}_2$ denoted H, a commitment to a message m using randomness r, r' is a tuple $(H^{r_1}m, G_0^r G_1^{r_1})$.

Sigma protocol for consistent commitments. Given a $\mathbb{G}_1, \mathbb{G}_2, G_0, G_1, H$, (c_1, c_2), (c_1', c_2') the prover shows that they know (r, r', r_1, r_1') such that $c_1/c_1' = H^r/H^{r'}$, $c_2 = G_0^r G_1^{r_1}$, and $c_2' = G_0^{r'} G_1^{r_1'}$.

1. Prover chooses (s, s', s_1, s_1') at random and computes $com_1 = H^s/H^{s'}$, $com_2 = G_0^s G_1^{s_1}$, and $com_3 = G_0^{s'} G_1^{s_1'}$ and returns (com_1, com_2, com_3).
2. Verifier sends a challenge e chosen at random in $\mathbb{Z}_N$.
3. Prover computes $t_1 := s+er$, $t_2 := s'+er'$, $t_3 := s_1+er_1$, and $t_4 := s_1'+er_1'$ and sends these to the verifier.
4. The verifier accepts if $com_1(c_1/c_1')^e = H^{t_1}/H^{t_2}$ and $com_2 c_2^e = G_0^{t_1} G_1^{t_3}$ and $com_3 {c'}_2^e = G_0^{t_2} G_1^{t_4}$.

The proof is straightforward and we omit it due to lack of space.

References

1. Abe, M., Haralambiev, K., Ohkubo, M.: Group to group commitments do not shrink. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 301–317. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_19
2. Adida, B.: Helios: web-based open-audit voting. In: van Oorschot, P.C. (ed.) USENIX Security Symposium, pp. 335–348. USENIX Association (2008)
3. Benaloh, J.C., Yung, M.: Distributing the power of a government to enhance the privacy of voters. In: Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing, pp. 52–62. ACM (1986)
4. Benaloh, J.C., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: Leighton, F.T., Goodrich, M.T. (eds.) STOC, pp. 544–553. ACM (1994)
5. Bernhard, D., Cortier, V., Galindo, D., Pereira, O., Warinschi, B.: SoK: a comprehensive analysis of game-based ballot privacy definitions. In: IEEE Symposium on Security and Privacy, pp. 499–516. IEEE Computer Society (2015)
6. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: pitfalls of the Fiat-Shamir heuristic and applications to helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_38
7. Chaum, D.: Untraceable mail, return addresses and digital pseudonyms. Commun. ACM **24**(2), 84–88 (1981)
8. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM **28**(10), 1030–1044 (1985)
9. Chaum, D.: Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In: Barstow, D., Brauer, W., Brinch Hansen, P., Gries, D., Luckham, D., Moler, C., Pnueli, A., Seegmüller, G., Stoer, J., Wirth, N., Günther, C.G. (eds.) EUROCRYPT 1988. LNCS, vol. 330, pp. 177–182. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-45961-8_15
10. Chaum, D., et al.: Scantegrity II: end-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In: EVT. USENIX Association (2008)
11. Cohen, J.D.: Improving privacy in cryptographic elections. Citeseer (1986)
12. Cohen, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme. In: FOCS, vol. 85, pp. 372–382 (1985)
13. Cuvelier, É., Pereira, O., Peters, T.: Election verifiability or ballot privacy: do we need to choose? In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 481–498. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40203-6_27
14. Demirel, D., Van De Graaf, J., Araújo, R.: Improving Helios with everlasting privacy towards the public. In: Proceedings of the 2012s international conference on Electronic Voting Technology/Workshop on Trustworthy Elections, p. 8. USENIX Association (2012)
15. Escala, A., Guasch, S., Herranz, J., Morillo, P.: Universal cast-as-intended verifiability. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 233–250. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_16
16. FIPS, P.: 186–4: Federal information processing standards publication. digital signature standard (DSS). Information Technology Laboratory, National Institute of Standards and Technology (NIST), Gaithersburg, MD, 20899–8900 (2013)

17. Guasch, S., Morillo, P.: How to challenge *and* cast your e-Vote. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 130–145. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_8
18. Haines, T., Gritti, C.: Improvements in everlasting privacy: efficient and secure zero knowledge proofs. Cryptology ePrint Archive, Report 2019/901 (2019)
19. Halderman, J.A., Teague, V.: The new south wales ivote system: security failures and verification flaws in a live online election. In: Haenni, R., Koenig, R.E., Wikström, D. (eds.) VOTELID 2015. LNCS, vol. 9269, pp. 35–53. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22270-7_3
20. Hazay, C., Mikkelsen, G.L., Rabin, T., Toft, T., Nicolosi, A.A.: Efficient RSA key generation and threshold paillier in the two-party setting. J. Cryptol. **32**(2), 265–323 (2019)
21. Karayumak, F., Olembo, M.M., Kauer, M., Volkamer, M.: Usability analysis of Helios - an open source verifiable remote electronic voting system. In: Shacham, H., Teague, V. (eds.) 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections, EVT/WOTE 2011, San Francisco, CA, USA, 8–9 August 2011. USENIX Association (2011)
22. Küsters, R., Truderung, T., Vogt, A.: Accountability: definition and relationship to verifiability. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, 4–8 October 2010, pp. 526–535. ACM (2010)
23. Locher, P., Haenni, R., Koenig, R.E.: Coercion-resistant internet voting with everlasting privacy. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 161–175. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_11
24. Merrit, M.: Cryptographic protocols. Ph.D. thesis (1983)
25. Moran, T., Naor, M.: Split-ballot voting: Everlasting privacy with distributed trust. ACM Trans. Inf. Syst. Secur. **13**(2), 16 (2010)
26. Nishide, T., Sakurai, K.: Distributed Paillier cryptosystem without trusted dealer. In: Chung, Y., Yung, M. (eds.) WISA 2010. LNCS, vol. 6513, pp. 44–60. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-17955-6_4
27. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
28. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
29. Ryan, P.Y.A., Rønne, P.B., Iovino, V.: Selene: voting with transparent verifiability and coercion-mitigation. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 176–192. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_12
30. Ryan, P.: A variant of the Chaum voter-verifiable scheme. In: Proceedings of the 2005 Workshop on Issues in the Theory of Security, pp. 81–88. ACM (2005)
31. Springall, D., et al.: Security analysis of the Estonian internet voting system. In: ACM Conference on Computer and Communications Security, pp. 703–715. ACM (2014)
32. Benaloh, J. Simple verifiable elections. USENIX Association (2006)
33. Wikström, D.: A commitment-consistent proof of a shuffle. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 407–421. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02620-1_28

34. Yang, N., Clark, J.: Practical governmental voting with unconditional integrity and privacy. In: Brenner, M., Rohloff, K., Bonneau, J., Miller, A., Ryan, P.Y.A., Teague, V., Bracciali, A., Sala, M., Pintore, F., Jakobsson, M. (eds.) FC 2017. LNCS, vol. 10323, pp. 434–449. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_27
35. Cramer, R., Franklin, M., Schoenmakers, B., Yung, M.: Multi-authority secret-ballot elections with linear work. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_7

Network Security

Using Partial Signatures in Intrusion Detection for Multipath TCP

Zeeshan Afzal(✉), Johan Garcia, Stefan Lindskog, and Anna Brunstrom

Karlstad University, Karlstad, Sweden
{zeeshan.afzal,johan.garcia,stefan.lindskog,anna.brunstrom}@kau.se

Abstract. Traditional security mechanisms such as signature based intrusion detection systems (IDSs) attempt to find a perfect match of a set of signatures in network traffic. Such IDSs depend on the availability of a complete application data stream. With emerging protocols such as Multipath TCP (MPTCP), this precondition cannot be ensured, resulting in false negatives and IDS evasion. On the other hand, if approximate signature matching is used instead in an IDS, a potentially high number of false positives make the detection impractical. In this paper, we show that, by using a specially tailored partial signature matcher and knowledge about MPTCP semantics, the Snort3 IDS can be empowered with partial signature detection. Additionally, we uncover the type of Snort3 rules suitable for the task of partial matching. Experimental results with these rules show a low false positive rate for benign traffic and high detection coverage for attack traffic.

1 Introduction

Intrusion detection systems (IDSs) are an integral part of network infrastructure protection. Signature based IDSs such as Snort3 [24] compare network traffic with signatures extracted from pre-defined rules. The comparison results in an alarm only when an exact and perfect match of a signature is found in the network traffic and other conditions stated in a rule are met. The detection is heavily dependent on the availability of a complete data stream. To ensure this, the Snort3 IDS caters for on-path fragmentation by using a TCP stream reassembly process that, if enabled, waits for attack patterns that are fragmented across multiple packets of the same connection. The process combines data from all packets in a stream into a larger pseudo packet and forwards it for detection [10]. Such a set-up has served its purpose well in the case of standard TCP. However, Snort3 cannot handle the case where the application data are fragmented across different paths (TCP connections).

Multipath TCP (MPTCP) [12] is an emerging protocol that is expected to overcome some inherent weaknesses in standard TCP. Originally designed for smartphones, MPTCP enables end-hosts to communicate using multiple paths at the same time for improved connectivity and/or a higher throughput. One of the implications of this is that many basic assumptions made by IDSs such as Snort3

A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 71–86, 2019.
https://doi.org/10.1007/978-3-030-35055-0_5

are no longer true [3,21]. The problem is two-fold. First, the security technologies that do not recognize MPTCP fail to perform their function even when the entire data stream is visible to them. For instance, assume that there is an IDS with a rule to detect a string "secret" in TCP packets. Using MPTCP, an adversary can fragment the signature and send "sec" and "ret" on two different paths. Consequently, the IDS will be matching the rule against one half of the string at a time instead of the whole string. It will not raise an alarm because neither "sec" nor "ret" match "secret". However, the full message "secret" will still arrive at its destination. This is also known as a cross-path data fragmentation. The built-in TCP reassembly is not adequate in this situation. Previous works [3,13] have addressed this problem where the task was relatively easier as the IDS had data from all paths and simply needed to correlate and reassemble it.

The second and more challenging problem for intrusion detection arises when the paths involved in an MPTCP connection are controlled by different Internet Service Providers (ISPs) and hence data from other paths are not available in an IDS. This limitation makes approaches such as distributed signature detection inapplicable, as there can be no sharing of information between the IDSs. In this scenario in which some data are missing, the IDS has to decide by itself whether an available data fragment is part of an actual signature. If the IDS uses exact matching, it will find no exact matches of any possible attack patterns. If general approximate matching is used, the number of matches will include far too many false positives, making the task of intrusion detection questionable by creating more problems than it solves [6].

This paper takes the first step towards detecting intrusion attempts caused by MPTCP cross-path data fragmentation and unavailability of data from other paths. In a first work of its kind, we investigate the extent to which an IDS that is operating independently can make reliable decisions. We also explore the bounds on the minimum data that an IDS needs to observe to make a decision. To this end, we formally present the attack model and employ a specially designed partial pattern matcher that enables an independently operating Snort3 IDS to reliably decide whether a data segment belongs to a signature in its database. Another contribution of this work is the discovery of Snort3 rules that are not suitable for partial signature detection. Leveraging the MPTCP protocol semantics and heuristics, we present a novel methodology that empowers Snort3 with partial signature detection with a low risk of false positives for suitable rules.

The rest of the paper is structured as follows. Section 2 provides the necessary background and summarizes related work. Section 3 describes the attack model and the partial signature matching algorithm. Section 4 discusses the detection set-up in more detail, explaining each building block. Section 5 provides a validation of the detection methodology. Finally, Sect. 6 presents an outlook and summarizes the work presented.

2 Background and Related Work

MPTCP [12] is an extension to TCP that enables a TCP connection to operate across multiple paths simultaneously, thus providing many performance and dependability advantages. An MPTCP connection consists of one or more subflows, where each of these subflows is a proper TCP connection but with additional MPTCP options. These options allow every subflow to be linked to an MPTCP connection. Once an MPTCP connection is established and multiple subflows are available, data can flow on any or all subflows simultaneously. The sender can utilize multiple paths to enhance bandwidth and ensure that the data are sent to the receiver as fast as possible using all available paths. See [12] for further details on MPTCP.

Smartphones were the main motivation behind the design and development of MPTCP, so the largest deployment of MPTCP is naturally on smartphones [9]. Using the two available wireless interfaces (cellular and WiFi) on smartphones, MPTCP enables a client to roam seamlessly without losing connectivity. For example, Siri in the Apple devices [7] benefits from MPTCP's handover capabilities by establishing an MPTCP connection with two subflows using the cellular and the WiFi interfaces. This deployment has not only significantly improved connectivity but also made Siri a lot faster. Another use case of MPTCP in smartphones has come in the form of combining the two available wireless interfaces for an increased throughput [8].

As most signature based IDSs use exact string matching to compare known attack patterns with network traffic, there is a strong possibility of evading them using variation(s) in attack patterns [15,17]. Since MPTCP allows an attacker to distribute attack patterns across multiple paths, this evades the exact string matching tasks on each individual path. This problem of detecting signatures from traffic where some parts are missing can be contrasted to the problem of approximate matching, which has been around for decades. Information theory introduced edit distance to measure how close or similar two strings are to each other. The similarity is measured in terms of how many operations are required to convert one string into the other. Levenshtein distance [18] is the most commonly used edit distance metric. Normally, it considers three operations: insertions, deletions, and substitutions. In computer science, Levenshtein distance can be used in approximate pattern matching [20]. Other metrics have been proposed in the literature such as Hamming distance [16] which only considers substitution operations when counting the number of operations needed to go from one string to another. Another metric known as Jaro-Winkler [25] distance considers transpositions only, whereas Longest common subsequence (LCS) [20] considers only two allowed operations, i.e., insertion and deletion.

As an alternative to unbounded approximate string matching, few works [11,22] have proposed constrained approximate string matching algorithms. These algorithms introduce constraints on the number of edit operations on the old attack patterns to ensure that false positives and negative rates are kept low. In previous work [2], we proposed an insert-only variation of the Levenshtein distance to enable accurate comparison of two strings for the case when edit operations such as deletions and substitutions do not appropriately model the process creating the differences in strings.

In terms of applied intrusion detection for detecting distributed signatures in MPTCP, to the best of our knowledge, only one work [19] exists. The authors proposed a distributed algorithm where each IDS coordinates and broadcasts its activities to all other IDSs. According to the authors, the proposed algorithm performs well, but optimization of the algorithm is required to reduce delay in detection when out of order packets are received. In addition, this approach assumes that the same organization owns parts of the paths where multiple IDSs are deployed or, alternatively, the IDSs are placed closer to end-points such that they can observe all the traffic. The approach will be infeasible for on-path IDSs where paths are owned or controlled by different ISPs and the traffic is fragmented using MPTCP across those paths.

3 Attack Model and Matching Algorithm

In this section, we describe the attack model considered in this paper. We also describe the matching algorithm to be used in detection of signatures from partial fragments.

3.1 Attack Model

We consider the case where there are multiple IDSs passively monitoring different paths into an organization's network. Our focus is on the most common use case of MPTCP, i.e., smartphones, where two paths are established and used over the two wireless interfaces. As shown in Fig. 1, these IDSs are working in offline mode and do not exchange messages or keep any joint state. They make decisions independently based on the network traffic that passes through them.

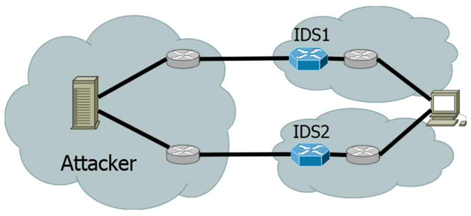


Fig. 1. Attack model.

We assume an attacker who uses MPTCP to distribute an attack pattern among multiple paths to perform arbitrary cross-path data fragmentation. This means that the attacker can alternate the path after every byte. To generalize, we consider the case of an organization with L Internet connections and IDSs operating on different autonomous systems (ASes). It can be shown that, with a length of M bytes for a particular signature, at least one of the IDSs must have observed a minimum detect length of at least $\lceil M/L \rceil$ bytes of the signature, if a message containing the signature has been transferred across the paths. This is true regardless of how an attacker might perform the splitting. Thus, the attack model followed in this paper assumes topological knowledge of its infrastructure by the organization and arbitrary fragmentation by the attacker, whereby the minimum detect length will be present on at least one of the paths.

3.2 Matching Algorithm

As noted earlier, approximate matching with the aim to detect cross-path fragmentation does not need to consider deletion or substitution operations, as the only possible transformation operation of a pattern string resulting from sending part of it on an alternate path is deletion. Therefore, the appropriate approximate matching operation is insertion. We have devised an MPTCP-aware multi-pattern approximate matching algorithm that uses the slice distance [2] metric suitable for approximate matching in cross-path fragmentation scenarios.

Formally, we consider a collection of V patterns where some pattern P is a string with length $M = |P|$. Furthermore, we receive indications that, in a data stream $D_s | s \in \{1, ..., z\}$, with a buffering of z bytes, there exists some number n of holes indexed by $j | j \in \{1, ..., n\}$. Each hole H_j starts at some position H_j^s in the data stream, and ends at some later position H_j^e. The high-level pseudo code for a matching algorithm over some sequence of data D_s and holes H_j is provided in Algorithm 1. The algorithm provides two operating cases, depending on the size of the hole H_j (i.e., $H_j^e - H_j^s$) as caused by the cross-path data fragmentation. Case 1 is where an attacker places packet boundaries in the middle of a pattern. Such attacks do not require repeated fragmentation into suspiciously small packet sizes, but will still break a non-multipath aware IDS matcher. The matcher now examines the region around the boundaries of the hole, i.e., the substrings $D_{\cdot, H_j^s}$ and $D_{H_j^e, \cdot}$ for potential matches against pattern substrings $P_{1,\cdot}$ and $P_{\cdot,|P|}$ using the count_matches method. Case 2 is where a pattern is split into multiple small pieces, and multiple holes may be present within one pattern by an attacker forcing multiple small fragmented packets to be produced, iteratively sending one or a few bytes on each of the paths. If required in an implementation context, different optimizations can be applied for the two different cases. For case 1, which corresponds to the multi-pattern general fix-constrained slice distance as discussed in [2], the Aho-Corasick algorithm [4] would be applicable, which would yield favorable computational complexity.

It can be noted that the pseudo code here focuses on ease of understanding and implementation rather than computational performance. The algorithm has a computational complexity of $O(V\bar{M}^2)$ per hole evaluation. While such complexity provides sufficient performance for our proof-of-concept implementation, we note that both cases can be sped up to a complexity of approximately $O(\bar{M})$ with the use of precomputed pattern match templates and rolling hashes, at a cost of increased memory complexity. As shown in [14], the achievable throughput of rolling hashes is above 2 Gbps, which is more than sufficient, as rolling hashing is only employed around packet fragment boundaries, i.e., holes. For the data set size of rules in Snort3, the resulting memory requirement of such an approach would be in the order of 100 MiB.

Algorithm 1. Partial signature matcher algorithm.

```
1: $R \leftarrow max(M)/2$
2: for $allholesH_j$ do
3:     if $H_j^e - H_j^s > R$ and $H_j^s - H_{j-1}^e > R$ and $H_{j+1}^s - H_j^e > R$ then
4:         for $P$ in $pattern\_collection$ do
5:             for $i$ in $roof(M/L)...M-1$ do
6:                 if i == count_matches($D_{H_j^s - i, H_j^e}$, $P_{1,i}$) then
7:                     return $P$
8:                 else if i == count_matches($D_{H_j^s, H_j^e + i}$, $P_{|P|-i,|P|}$) then
9:                     return $P$
10:                end if
11:            end for
12:        end for
13:    else
14:        for $P$ in $pattern\_collection$ do
15:            for $i$ in $1...M-1$ do
16:                if $|P|$ == count_matches($D_{H_1^s - i, H_1^e}$, $P_{1,|P|}$) + $sum(hole\_sizes)$ then
17:                    return $P$
18:                end if
19:            end for
20:        end for
21:    end if
22: end for
```

4 Detection Methodology

This section describes the detection of partial signatures using Snort3 in an MPTCP setting. Figure 2 shows the different parts of the detection methodology. The three blocks in bold are proposed by us, while the remaining are part of Snort3 by default and are used with no changes. As shown in the figure, network packets are processed by up to four different processes before being passed to the Snort3 Detection engine. As packets arrive, Snort3 performs basic protocol decoding and passes only TCP packets to the MPTCP inspector while the remaining packets are forwarded to the Detection engine. The inspector forwards necessary packet information to the MPTCP reassembler that performs the stream reassembly and also detects possible missing segments. Depending on the response from the MPTCP reassembler, the recreated data stream is either passed on to the Partial matcher if the response suggests missing segments or is directly forwarded to the Detection engine if the response suggests no missing segments. It should be noted here that the decision as to whether a packet or flow triggers a Snort3 rule and creates an alarm is entirely left to the Snort3 Detection engine, which is the ultimate decision maker. The role of the processes that precede the Detection engine is only to help it make that decision accurately. Below, we discuss the inner working and details of each of the processes involved in the methodology.

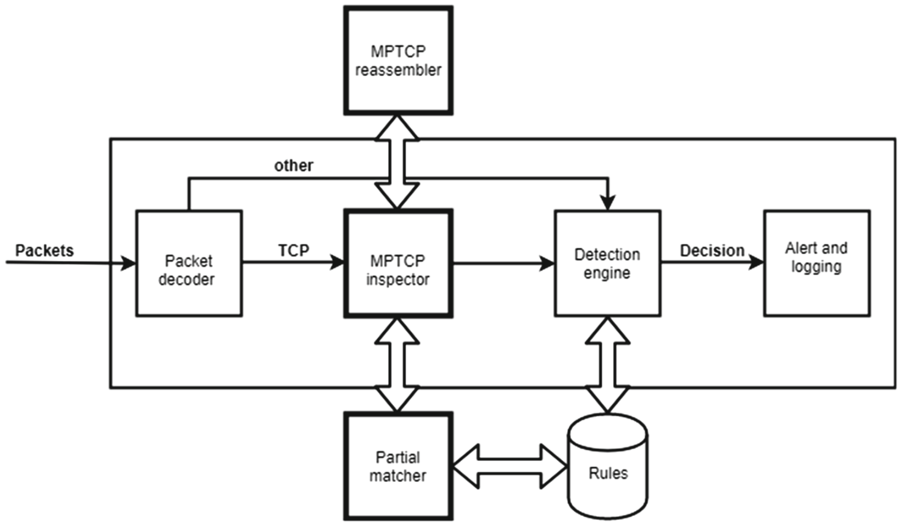


Fig. 2. Flow diagram of proposed detection methodology.

4.1 Packet Decoder, Detection Engine, and Logger

These are the three main systems that make up standard Snort3 and are used with no modifications in our methodology. The Packet decoder is the system that receives raw traffic and performs protocol decoding. It sets different pointers, e.g., location of data and size of payload, in the packet data to help the Detection engine in later analysis. The Packet decoder also translates unsupported encoding to an understandable format.

The Detection engine is the system that performs signature detection in Snort3 using a set of rules. This is done by dividing the rules in a two-dimensional linked-list. Packets on a particular 5-tuple are only matched to relevant signatures for that 5-tuple. If the Detection engine detects a signature and other options in a rule are true, then the Alert and logging system is used to log the alarm.

4.2 MPTCP Inspector

Snort3 [23] is the third generation version of the popular Snort IDS. One of the main motivations behind its development was making Snort's detection components extendable. Through the use of dynamic inspectors, it is possible to extend

Snort3 beyond its basic capability. Conceptually, once the Packet decoder system has performed decoding of packets, packets can be passed to any requesting preprocessor or inspector instead of the Detection engine. An inspector can perform a wide range of tasks, as custom code written in C++ can be executed here.

We have developed an MPTCP inspector to request all TCP packets from the Packet decoder system. The inspector filters received packets further by only processing the TCP packets with MPTCP options and passes the remaining TCP packets directly to the Detection engine. For every MPTCP data packet (referred to as original packet in the subsequent discussion), the MPTCP inspector extracts and passes the packet header and packet data to the MPTCP reassembler module and waits for the response. The response from the reassembler decides what happens next. If the reassembler observed a previous data packet for the same MPTCP connection, it reassembles the data stream according to the data sequence numbers and sends it back to the MPTCP inspector. If it detects a missing segment(s), this is also flagged to the MPTCP inspector.

If the response from the MPTCP reassembler shows that no missing data segment(s) was detected, the MPTCP inspector replaces the packet payload of the original packet with the reassembled payload, updates the packet length, and sends the packet to the Detection engine. However, if the reassembler flags a missing segment(s), the inspector utilizes the Partial matcher system to ensure that a partial fragment of a signature cannot evade detection. In that case, the reassembled payload and other information from the packet header such as the source and destination addresses and ports is forwarded to the Partial matcher. Any possible matches of the data stream in the matcher are returned. In case of a match, a new pseudo packet is generated with the same packet header as the original packet but with the updated length. The packet payload is replaced by the match returned by the pattern matcher. This packet is then forwarded to the Detection engine. In case of multiple matches by the Partial matcher, multiple pseudo packets can be created with different packet payloads and lengths and passed on to the Detection engine. The possible generation of pseudo packets by the MPTCP assembler is akin to the default TCP stream reassembly process in Snort3, as discussed earlier.

4.3 MPTCP Reassembler

The MPTCP reassembler is an enhancement of a tool [13] implemented in Python3 to perform MPTCP data stream reassembly to prevent cross-path data fragmentation attacks [3]. The MPTCP inspector forwards all MPTCP packets to this module. The reassembler keeps track of all MPTCP connections and their subflows using a buffer and tags every connection by a unique token. The handshake packets (both the initial handshake and subflow handshake(s)) are used to create entries in a dictionary data structure. On the reception of a data packet, data are saved for that particular connection in a buffer by looking it up in the dictionary. At the same time, the reassembly process kicks in and reassembles the data stream of that connection in the correct order by going through all subflows and using MPTCP sequence numbers. The reassembled data stream is sent back to the MPTCP inspector. The current implementation of the reassembler is a proof-of-concept and has aspects that could be improved. For example, with the arrival of each new MPTCP data packet, the reassembled stream grows in size as it contains data from the current data packet as well as all previous packets on that MPTCP connection. This can be a problem for production systems where timely flushing of the buffered data should instead be implemented.

Apart from the reassembly functionality, the MPTCP reassembler performs another important task of identifying missing segments while reassembling datastreams. This is done using the Data Sequence Signal (DSS) option in MPTCP that contains data sequence numbers and the datal-level length. A data packet can be missing in a stream because that packet could be routed over a different path to the destination. The reassembler identifies that and flags it to the MPTCP inspector. Additionally, a timer is associated with each MPTCP connection based on the last observed packet for it. If the next packet for that connection does not arrive before the timer expires and no connection closure packets are observed, this is also flagged to the MPTCP inspector to warn it of a possible missing segment(s) so that the Partial matcher can be utilized. We use a timer set to 30 s for our proof-of-concept. This will need to be tuned accordingly for a production system.

4.4 Partial Matcher and Rules

An implementation of the algorithm described in Sect. 3.2 is used in the detection methodology. The implementation uses Python NumPy arrays to store all incoming data. As Snort3 rules can define signature patterns in different formats, i.e., text, hex, or a mix of both, these patterns are all converted to a consistent integer format and stored in NumPy arrays. The incoming data stream is also saved in a consistent data type of integer arrays. Storing all data in this manner allows for NumPy array matching operations to be used directly. By employing our tailored matching algorithm, we intend to reduce the number of possible matches by only considering the ones that are possible by making insertions into the observed data. We also take a number of additional steps to further reduce the number of matches. In the case of Snort3 rules, each rule defines a 5-tuple in its header. This 5-tuple consists of the protocol and the IP addresses/ports of the sender and the receiver, respectively. The signature in the rule options is only matched for that particular 5-tuple. Using the same logic, the Partial matcher in our detection methodology reduces the number of possible signatures that a particular packet can trigger. Only the relevant signatures for the observed packet are considered by the Partial matcher.

5 Evaluation

In this section, we describe how the detection methodology discussed in Sect. 4 is evaluated. We experiment with two different datasets to evaluate it thoroughly.

5.1 Datasets

This subsection describes the datasets used to evaluate the methodology. For the proposed detection logic to be effective, it has to perform well with both the normal (benign) traffic as well as the attack traffic.

Normal Traffic. For normal traffic with no attack signatures, the methodology should generate no or very few false positives. Although millions of Apple devices use MPTCP, there are not many MPTCP servers on the Internet to date. This makes collection of real MPTCP traffic a problem. To overcome this hurdle, we set-up an MPTCP SOCKS5 proxy as depicted in Fig. 3. An MPTCP-capable client uses shadowsocks to connect to a remote shadowsocks server and forwards its traffic to it. The shadowsocks server supports MPTCP for all TCP connections. It communicates with the Internet over TCP as requested and sends the responses back to the client. For the client, the entire process is transparent and it seems as though it communicated with an MPTCP server. We configured MPTCP such that 2 subflows are created for each MPTCP connection. For scheduling packets over the subflows, a round-robin scheduler with default parameters is used. The scheduler ensures that data are sent on both subflows in a round-robin fashion.

A total of 17 different HTTP based websites were browsed on the Internet to collect a trace of 151,367 packets. Eleven of the selected websites are amongst the top 100 websites in the world according to Alexa [5]. The rest are local European websites, including the websites of municipalities of cities. The trace consists of 900 MPTCP connections, each with 2 MPTCP subflows making up a total of 1800 MPTCP subflows, and we assume that it is benign. The reason for collecting HTTP traffic (port 80) is that a signature-based IDS requires plaintext application data. We realize this limitation of signature-based IDSs, but observe that these IDSs are still common and will continue to be used in a number of controlled settings.

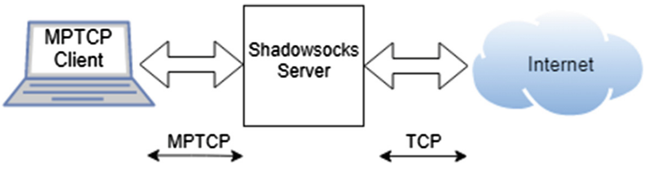


Fig. 3. MPTCP proxy set-up for traffic collection.

Attack Traffic. We generate attack traffic synthetically from Snort3 rules to counter the lack of a realistic IDS testing attack dataset. Our approach which was previously presented in [1] uses an interpreter to translate each Snort3 signature into a corresponding payload. Furthermore, to mimic the data fragmentation effect of MPTCP, fragments are derived for each payload. These fragments are ensured to be exactly half of the original payload in length (or as close as possible).

Each resulting payload fragment is small enough to be encapsulated into a packet. These packets are then sent to mimic an MPTCP connection with two subflows, i.e., two paths and an attacker that equally distributes the data stream among the two paths.

5.2 Snort Rules

The official Snort3 rules (snapshot 2990) are used in this work. The ruleset consists of 26808 TCP rules divided into over 60 rule categories. For evaluation, we filter the rules by destination port and only consider ports that have at least 100 rules. Additionally, we filter out rules that use regular expressions in Perl as they are limited in number. In total, 23320 rules divided among seven rule categories are identified and used.

5.3 Results and Discussion

This subsection presents the evaluation results generated using the methodology discussed in Sect. 4. During the normal operation of an IDS, it spends most of the time processing non-attack traffic [6]. It is therefore vital for the detection methodology to not generate too many false positives for it to be useful. In the design of the methodology, we have taken a number of steps to ensure the same. First and foremost, only MPTCP traffic is processed. The non-MPTCP traffic is directly forwarded to the Detection engine with no modification. Secondly, the usage of Partial matcher is only enabled in special conditions, i.e., when the MPTCP reassembler detects missing segment(s). Lastly, the Partial matcher itself is designed in a way such as to ignore matches that are not possible and only consider matches possible due to missing bytes. This combination of precautionary steps ensures that partial matching is performed only when it is absolutely necessary and in a way that keeps false positives as low as possible. To evaluate how well these ideas work in practice, we experiment with the captured traffic.

Normal Traffic. As we are only focused on those false positives that are actually caused by our Partial matcher, we test the benign traffic twice. First, we consider the trace as a whole where all MPTCP connections and their respective subflows have no data missing. Since there are no missing segments, the Partial matcher is not used and detection is performed by the Detection engine directly. We noticed a total of only 3 false positives in this case. These false positives represent the case where Snort3 wrongly classifies benign HTTP traffic as malicious.

Secondly, to imitate an MPTCP scenario with only partial traffic available, we split the big trace into separate pcap files (1800 in total), one for each subflow, and test them individually. In this case, where the Partial matcher is used, an initial analysis showed a high number of 867 false positives. Upon further analysis, we discovered that 846 of them are triggered by a single generic pattern "HTTP/1.1 200" that is common in web traffic. The Snort3 rule from which this pattern comes aims to detect invalid HTTP headers by defining a specific location in the packet payload for this pattern and a condition that a second pattern "OK" should not be present after the first pattern. This second condition is not compatible with our methodology, as we only attempt to identify the existence of certain patterns. Thus, such Snort3 rules that rely on the absence of certain patterns should not be used for partial detection of signatures. This may lead to some false negatives but we identified a total of only 91 such rules (referred to as unsuitable rules). The remaining 21 false positives are caused by patterns that are short in length, and their rules specify other information, such as pattern location, that is not available to the Partial matcher. A comparison of the two log files showed that only 18 false positives are unique to the partial matching case. This validates the detection methodology as the usage of the Partial matcher does not generate a very large number of additional false positives.

Attack Traffic. Table 1 below summarizes the results for attack traffic according to the top Snort3 rules' destination ports. For each category, all attacks are detected, i.e., 100% attack detection coverage. The difference in detection lies in the frequency of matching each signature fragment with only one signature (defined as unique detection ratio in the table). From the table, it can be seen that the largest category of rules is "any", as these rules define no destination port. For this category of rules, using the generated attack traffic and the detection methodology, it is possible to detect 80.4% of the signatures with no risk of multiple matches (multi-matches). We define a multi-match as an instance when a signature fragment matches with more signatures than just the intended signature. For the remaining 19.6% rules from this category, the Partial matcher generates multi-matches as it has no means to further filter and provide more accurate results based on the input fragment it sees. As an example of a multi-match, for a signature fragment "User-Agent: Mozilla/5.0", the Partial matcher gives two matches; "User-Agent: Mozilla/5.0(Linux)" and "User-Agent: Mozilla/5.0(Android)". Considering only the input fragment, it cannot do better. During detection, such situations are handled by crafting multiple packets in the MPTCP inspector and passing them to the Detection engine. The idea here is to rely on other details in the Snort3 rules to let the Detection engine trigger an alarm for only one of the packets. In the worst case, when the rule header and options provide no means for the Detection engine to know which of the possible rules to trigger, there will be a possibility of multiple alarms being generated. We believe that these cases can be further reduced by optimizing the Snort3 rules. In particular, rules that are for the same 5-tuple and have patterns that are very similar to each other need optimizations to help avoid triggering multiple alarms.

For the rest of the categories in the table, the ratio of unique detection (with no multi-matches) varies from 83.7% for port 25 to 43.2% for ORACLE port rules. Three types of rule categories dominate the number of rules. These include "any", "HTTP", and "port 25". These three categories account for more than 84% of Snort3 rules in snapshot 2990. In all, we show that it is possible to detect signatures from their partial fragments with no multi-matches for over 76% of the Snort3 rules in snapshot 2990.

Table 1. Evaluation of the detection methodology using attack traffic.

Port category	Total attacks	Detected attacks	Single matches	Multiple matches	Unique detection ratio	Attack detection coverage
Port 139/445	62	62	43	19	69.3%	100%
Port 21	125	125	73	52	58.4%	100%
Port 443	116	116	66	50	56.8%	100%
ORACLE	335	335	145	190	43.2%	100%
Port 25	4831	4831	4046	785	83.7%	100%
HTTP	8557	8557	5970	2587	69.7%	100%
any	9294	9294	7474	1820	80.4%	100%
Overall	**23320**	**23320**	**17817**	**5503**	**76.4%**	**100%**

To validate and support that the Partial matcher is designed in a way to keep false matches down to a minimum, we compare it against a well known approximate string matching algorithm, i.e., Levenshtein distance [18]. For this task, the Partial matcher from the methodology shown in Fig. 2 is replaced by an implementation of general purpose Levenshtein distance. All other parameters including the bounds on the minimum number of matches required are kept the same. It can be seen from Table 2 that the number of multi-matches increases and the number of single matches decreases for each port category, when Levenshtein distance is used on the attack traffic. This was due to the fact that Levenshtein distance considers all three types of edit operations. This results in it matching unrelated fragments at times and missing possible matches at other times. When normal and benign traffic is instead used, a large number of 822 false positives are generated (considering only suitable rules), in comparison to only 21 false positives generated by the proposed Partial matcher as shown before. These results validate the ideas of the paper, as the use of any other alternative than the specially tailored Partial matcher when matching would create many more multi-matches for the attack traffic, as shown by the low 20.8% unique detection ratio and a high number of false positives for the normal traffic.

Table 2. Results of using Levenshtein distance with attack traffic.

Port category	Single matches	Multiple matches	Unique detection ratio
Port 139/445	1	98	0%
Port 21	18	106	14.5%
Port 443	32	75	29.9%
ORACLE	38	324	10.4%
Port 25	1208	2706	30.8%
HTTP	579	7866	6.8%
any	2557	5681	31.0%
Overall	**4433**	**16856**	**20.8%**

6 Outlook and Concluding Remarks

IDSs that perform exact string matching malfunction in the presence of data stream fragmentation enabled by protocols such as MPTCP. Existing solutions either rely on communication between different IDSs or the application of general approximate string matching. The first approach is not always practical, as the data required for signature matching need to be shared among different paths that can be controlled by different ISPs. The second approach of general approximate string matching will likely yield too many false positives. This calls for a novel approach to detect intrusions from partial traffic.

We have therefore in this paper proposed a way to detect signatures from partial fragments using Snort3. The detection logic relies on a specially tailored Partial matcher and makes use of MPTCP protocol semantics and heuristics. Our experiments show that the detection methodology is effective. For attack traffic based on Snort3 rules, all attacks are successfully detected with a 100% detection coverage where over 76% signatures can be detected with no risk of multi-matching. The evaluation using the collected benign traffic shows that, if the traffic does not contain any attack signatures, then the detection remains mostly silent and does not generate too many false positives for the suitable Snort3 rules. Identification of such rules that are suitable for partial matching is another contribution of this work. It should be noted that the traffic samples used in our evaluations are relatively small and should hence be regarded as a proof-of-concept. A more extensive evaluation should be conducted before implementation in a production system. We envision a number of steps to take this work forward. This involves a comprehensive evaluation using more realistic attack traffic and further enhancements of the detection methodology. For instance, instead of separate processes for MPTCP stream reassembly and partial matching, their functionality could be implemented within the MPTCP inspector. The unique detection ratio could also be further improved using packet meta-data information such as packet size. The code of the current version of our work is freely available[1].

Acknowledgment. The work is being carried out in the High Quality Networked Services in a Mobile World project funded partly by the Knowledge Foundation of Sweden. The authors are grateful to Henry Foster for providing the building blocks required for this work.

References

1. Afzal, Z., Lindskog, S.: IDS rule management made easy. In: 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Piteşti, Romania (2016)
2. Afzal, Z., Garcia, J., Lindskog, S., Brunstrom, A.: Slice distance: an insert-only Levenshtein distance with a focus on security applications. In: 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018, pp. 1–5 (2018)
3. Afzal, Z., Lindskog, S.: Multipath TCP IDS evasion and mitigation. In: Lopez, J., Mitchell, C.J. (eds.) ISC 2015. LNCS, vol. 9290, pp. 265–282. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23318-5_15
4. Aho, A.V., Corasick, M.J.: Efficient string matching: an aid to bibliographic search. Commun. ACM **18**(6), 333–340 (1975)
5. Alexa: The top 500 sites on the web (2019). https://www.alexa.com/topsites
6. Axelsson, S.: The base-rate fallacy and the difficulty of intrusion detection. ACM Trans. Inf. Syst. Secur. **3**(3), 186–205 (2000)
7. Bonaventure, O.: Apple seems to also believe in Multipath TCP (2013). http://perso.uclouvain.be/olivier.bonaventure/blog/html/2013/09/18/mptcp.html
8. Bonaventure, O.: In Korea, Multipath TCP is pronounced GIGA path (2015). http://blog.multipath-tcp.org/blog/html/2015/07/24/korea.html
9. Bonaventure, O., Seo, S.: Multipath TCP deployments (2016). https://www.ietfjournal.org/multipath-tcp-deployments/
10. Caswell, B., Beale, J.: Snort 2.1 Intrusion Detection. Elsevier, Amsterdam (2004)
11. Chitrakar, A.S., Petrovic, S.: Constrained row-based bit-parallel search in intrusion detection. In: 11th Norwegian Information Security Conference (NISK), Bergen, Norway, November 28–30 (2016)
12. Ford, A., Raiciu, C., Handley, M., Bonaventure, O., Paasch, C.: TCP extensions for multipath operation with multiple addresses. Standards Track RFC 6824 (2019)
13. Foster, H.A.: Why does MPTCP have to make things so complicated? (2016). https://calhoun.nps.edu/handle/10945/50546
14. Garcia, J.: A fragment hashing approach for scalable and cloud-aware network file detection. In: 9th IFIP International Conference on New Technologies, Mobility and Security, (NTMS), Paris, France, 26–28 February 2018, pp. 1–5 (2018)
15. Ptacek, T.H., Newsham, T.N., Simpson, H.: Insertion, evasion, and denial of service: eluding network intrusion detection. Secure Networks (1999)
16. Hamming, R.W.: Error detecting and error correcting codes. Bell Labs Tech. J. **29**(2), 147–160 (1950)

[1] Source code available at https://github.com/randomsecguy/.

17. Jingping, J., Kehua, C., Jia, C., Dengwen, Z., Wei, M.: Detection and recognition of atomic evasions against network intrusion detection/prevention systems. IEEE Access **7**, 87816–87826 (2019)
18. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Sov. Phys. Dokl. **10**, 707–710 (1966)
19. Ma, J., Le, F., Russo, A., Lobo, J.: Detecting distributed signature-based intrusion: the case of multi-path routing attacks. In: IEEE Conference on Computer Communications (INFOCOM) 2015, Kowloon, Hong Kong, 26 April–1 May 2015, pp. 558–566 (2015)
20. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
21. Pearce, C., Zeadally, S.: Ancillary impacts of Multipath TCP on current and future network security. IEEE Internet Comput. **19**(5), 58–65 (2015)
22. Petrovic, S.V., Golic, J.D.: String editing under a combination of constraints. Inf. Sci. **74**(1–2), 151–163 (1993)
23. Russ Combs: Project Snort++, a.k.a. snort 3.0. https://blog.snort.org/2014/12/project-snort-aka-snort-30.html
24. The Snort Team: Snort official website. https://www.snort.org/
25. Winkler, W.E.: String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In: Proceedings of the Section on Survey Research, pp. 354–359 (1990)

Analysis of Topology Poisoning Attacks in Software-Defined Networking

Thanh Bui(✉), Markku Antikainen, and Tuomas Aura

Aalto University, Espoo, Finland
thanh.bui@aalto.fi

Abstract. In software-defined networking (SDN), routing decisions are made by a trusted network controller, which communicates with each forwarding device over a secure control channel. While this architecture avoids many security issues of distributed routing protocols, SDN remains vulnerable to topology poisoning attacks during topology discovery. Faked link information can cause wrong routing decisions by the controller and, thus, enable the attacker to reroute some traffic flows to compromised nodes. This paper provides both qualitative and quantitative analysis of topology poisoning attacks in SDN. We classify the attacks including new variants and analyze how their impact depends on the network topology, routing policy, and attacker location. While most of the literature emphasizes the security of the SDN controller and control channels, we assume them to be secure and aim to understand the ability of a small number of compromised switches to divert traffic flows. This is important because the low-cost, heterogeneous network equipment available for SDN may not be entirely trusted and because targeted attacks often start from the compromise of a single device.

1 Introduction

Software-defined networking (SDN) [19] separates the control plane of the network from the data plane and moves it to a software-based controller, which communicates directly with the switches. This paper investigates the vulnerability of SDN to one particular class of attacks: topology poisoning. In traditional networks, an attacker can spoof or tamper with routing advertisements to manipulate how the control-plane elements view the network topology [23]. Compared to traditional networks, a well-managed SDN is relatively resistant to such attacks because the routing decisions are not made by a distributed protocol but by a trusted controller, which communicates with each network device directly, over a secure channel. Nevertheless, it has been discovered that the network topology service of SDN controllers can be poisoned by compromised nodes [3,9,14]. In these attacks, the attacker creates fake links into the network so that the controller, which typically uses a shortest-path algorithm to decide on the routes, sends more data flows through the compromised nodes. Until now, however, the attacks by *compromised switches* in SDN have been poorly understood and analyzed. We aim to shed light into this area.

A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 87–102, 2019.
https://doi.org/10.1007/978-3-030-35055-0_6

Attacks mounted by a small number of compromised nodes in an otherwise secure network are a timely issue for two reasons. First, in targeted attacks against high-security organizations, the attacker first tries to gain any kind of small foothold, such as one compromised network device, and then starts to extend this gradually. Second, the trustworthiness of network equipment is often called into question, for the fear of both targeted trojans and mass-surveillance functionality in the devices. This concern becomes even more acute with SDN because one of the promises of SDN is that networks can be built with low-cost, heterogeneous equipment that only needs to implement the standard interfaces.

We analyze the topology poisoning attacks against SDN both qualitatively, to understand the principles, and quantitatively with simulations, to assess and explain their impact. Our focus is specifically on attacks mounted from a small number of compromised SDN switches. Moreover, we focus on types of topology poisoning attacks that previous research has found particularly difficult to defend against [2,9,14]. This is also reflected in the defensive strategies that we discuss, which are more on how the effect of these attacks can be mitigated, rather than trying to completely prevent them. For concreteness, we use OpenFlow [22] as the example protocol and architecture. The ideas and results are, however, general and apply to other SDN technologies.

2 Background

This section gives an overview of OpenFlow controller channel and topology discovery as well as related work.

2.1 OpenFlow Controller Channel and Topology Discovery

OpenFlow [20] is the most common *controller channel* protocol used by SDN network controllers to manage the switches. Routing rules and security policies are implemented in software at the controller and in the SDN applications on top of it. The controller then installs the necessary flow table entries to the switches, which forward packets based on them.

In order to make routing decisions, the controller needs to know the network topology, i.e. how the switches are connected to each other. In SDN, the old distributed layer-2 and layer-3 topology discovery and management techniques (e.g. spanning-tree protocol) are replaced by new, controller-driven mechanisms. The OpenFlow controller typically uses the OpenFlow Discovery Protocol (OFDP), which leverages the Link Layer Discovery Protocol (LLDP) packets [1], to dynamically detect layer-2 links between adjacent OpenFlow switches.

The link discovery process is as follows. First, the controller sends to one switch an OFPT_PACKET_OUT message, which contains an LLDP packet and the instruction to forward it on a specific port. The switch forwards the LLDP packet on the specified port. If there is another switch connected to the port, it receives the LLDP packet and sends it to the controller in an OFPT_PACKET_IN message along with the ingress-port identifier. The controller can thus reason that there is a directional link between the two switches. The controller then proceeds to do the same in the other direction.

2.2 Related Work

Topology poisoning is a well-known problem in traditional networks where the attacks mainly exploit distributed routing protocols [15,23,24,29]. The opportunities for topology poisoning in SDN are narrower than in traditional networks. In a well-managed SDN network, there is a direct secure channel between the non-compromised switches and the controller. The switches do not aggregate routes or forward unauthenticated topology or routing information from each other. Thus, the compromised switches cannot spoof or make malicious modifications to the routes or to the forwarded topology data. This means that the only way for the compromised switches to influence routing in SDN is to tweak the topology-discovery process locally, between the compromised switches and their neighbors.

The earlier literature on SDN security has focused on the security of SDN applications [26,28,30], real-time verification of network policies [13,17,18,25], and vulnerabilities in the controllers [20,21,27] as well as controller-switch communication [5,9]. Much less effort has been put into studying attacks originating from the data plane and, more specifically, from compromised nodes [3,4,16].

Topology poisoning in SDN has stimulated some interest from scholars. Hong et al. [14] presented several ways in which the controller's network view can be poisoned by a compromised host even when the controller channel is protected with TLS. The proposed attacks are carried out either by creating fake Link Layer Discovery Protocol (LLDP) packets or by forwarding genuine ones from one switch to another. We build on this basic technique while discussing a broader range of attack variants and analyzing them in more depth.

Hong et al. proposed adding an extra authenticator to LLDP packets, but acknowledge that it does not prevent the relaying of genuine packets. Similar authentication mechanism was also proposed by Alharbi et al. [2]. There are also defense strategies for topology poisoning that are based on anomaly detection [6,9]. These, however, are limited to some attack scenarios such as where the attacker does not actively try to avert detection or where only a single network element has been compromised.

3 Topology Poisoning Attacks

This section presents our threat model and different variants of topology poisoning attacks in SDN.

3.1 Threat Model

In this paper, we assume that the SDN controller and the control channels are secure. However, the attacker is in control of a small number of *compromised switches*. The goal of the attacker is to poison the controller's view of the network topology and route more traffic to the compromised nodes.

By a compromised switch, we mean that the attacker is able to access and manipulate the switch's configuration, authentication credentials, and flow tables. The compromised switch is not assumed to have any special hardware capabilities. For example, the switch could be compromised by accessing it via a console serial port, by guessing the administrator password, or by exploiting a software vulnerability. To implement a persistent compromise of the switch, the simplest way would be to add a second, *malicious controller* to it (multiple controllers is an OpenFlow 1.2+ feature). All the topology-poisoning attacks considered in the paper can be achieved with this method.

The attack happens in the topology discovery phase. The attacker spoofs fake links that create shorter or alternative paths in the network. Since all routing algorithms favor shorter paths at least to some extent, the SDN controller will route packets via these fake links. The attacker can then sniff the packets or mount MitM attacks on the data plane.

3.2 Attack Principle

The principle of the topology poisoning attacks is that the attacker first manipulates the propagation of the LLDP packets to fabricate non-existing links and then establishes tunnels that make these links appear functional. For example, if there are two compromised switches, whenever one of them receives an LLDP packet that should cross the fake link, it tunnels the packet to the other switch, which encapsulates it into an OFPT_PACKET_IN message and sends it to the controller. As the result, the controller thinks that there is a direct link between the two switches.

Apart from this basic principle, the attacker needs to avoid creating a forwarding loop. Consider the situation where one tunnel endpoint, i.e. one of the compromised switches, wants to send a tunnel packet to the other. It has to forward the tunnel packet to an adjacent good switch first. If the packet is addressed to the other tunnel endpoint, the adjacent switch will usually return the packet right back to the compromised switch because the controller thinks that the shortest route is through the fake link. To avoid this pitfall, the two tunnel endpoints need to communicate via a third node, called the *relay node*. To set up the relay node, the attacker could compromise a host in the network, rent a virtual machine if it is a data center, or use a remote host in the Internet. However, the relay node cannot be just anywhere on the network. It needs to be located in such a way that the packets from the tunnel endpoints to it will not be routed into the tunnel. We will discuss this requirement further in Sect. 5.4.

3.3 Attack Variants

In this section, we present four different variants in which the fake links and tunnels can be set up for the attacks. Among these, only the basic variant has been discussed in the literature [3,9].

Basic Variant. In the most obvious variant of the attack, the controller is fooled into thinking that there is a direct link between two compromised switches.

Just one fake link will be created for one pair of compromised nodes. The apparent distance between two good nodes that are connected through this link will be at least 3 hops. If there are n compromised nodes, then the number of fake links can grow to $n(n-1)/2$.

Neighbor Variant. This variant also requires two compromised switches, but the fake link is created between two good switches that are neighbors to the two compromised switches. The compromised switches become invisible and tunnel packets transparently between the neighbors. They do not participate in the link-discovery process in any other way.

The compromised switches typically have many neighbors, and it is possible to pair them up to create multiple tunnels. However, one neighbor needs to be sacrificed for the tunnel implementation. If the degree of both compromised nodes is k, there can be up to $k-1$ fake links via each of them. With n compromised nodes of degree k, the number of fake links can be as high as $\lfloor \frac{n}{2}(k-1) \rfloor$. The minimum apparent length of the paths that traverse the fake link is just 1. The greater reduction in path lengths and the larger number of apparent links make this variant attack worth investigating in comparison to the previous one.

Merging Variant. In this attack, two or more compromised switches merge together and appear as one big virtual switch with many neighbors. They all share one identity, which can be taken from one of the participating switches.

With two compromised nodes of degree k, this variant attack creates fake paths between all the $k(k-1)$ pairs of neighboring good switches (reserving one neighbor for the tunnel on the switch that assumes the identity of the other). When the number of compromised switches grows to n, the number of such fake paths becomes is as large as $(k-1)^2 \cdot (n-1)(n-2)/2 \; + \; k \cdot (k-1)(n-1)$. The minimum apparent length of the paths that make use of these fake routes is 2 hops. While the resulting path lengths are one hop higher than in the previous variant, this third variant becomes interesting when $n > 2$ or k is large because the number of good-neighbor pairs connected via the fake links grows faster.

Single-Switch Variant. The last variant attack differs from the previous ones in that only one compromised switch is needed. The fake links are established between two neighbors of the same compromised switch, and the compromised switch itself becomes transparent. No tunnel or relay is used. Instead, the attacker configures the compromised switch to forward the LLDP packets received from one port directly to another port.

If the compromised switch directly connects to k good switches, the attacker can fabricate a maximum of $\lfloor \frac{k}{2} \rfloor$ fake links between these switches. The minimum apparent distance between good nodes through the fake link is just 1. While this attack causes only minor changes to the apparent topology of the network (shortening some paths by 1), it is worth investigating because it can be launched locally at a single compromised switch without any collusion or coordination.

3.4 Attack Implementation

To verify that the topology poisoning attacks described above work against real SDN technology, we implemented them in an emulated network environment. Table 1 summarizes the software and methods used in the experiments.

Table 1. Implementation environment

Network emulator	Mininet 2.2.0
Switches	OpenvSwitch 2.3.1 on Mininet hosts
Controller	Floodlight 1.0
Routing algorithm	Shortest-path routing
Southbound protocol	OpenFlow version 1.0
Malicious controller	POX

In the experiments, the malicious switches were controlled by a malicious controller. The malicious controller also acted as the relay node for the tunnels. Moreover, the control channel between the good controller and the compromised switches was redirected from the compromised switch to the malicious controller. This was done with NAT-like address rewriting rules in the compromised switch's flow table. The malicious controller then emulated the behavior of good switches and spoofed the state of the compromised switches to the good controller so that they appeared to be functioning as normal. It is important to emphasize that the compromised switches were standard OpenFlow 1.0 switches with no software modifications. They were compromised only in the sense that the attacker could change their controller and could access their authentication credentials for spoofing the OpenFlow channel towards the good controller.

4 Attack Simulation

In this section, we assess the threats of the topology poisoning attacks that are described in the previous section in simulated networks. Our goal is to analyze topology spoofing on a general level. Thus, rather than picking one specific network and routing algorithm for the simulations, we assess the threats against a wide variety of network topologies and different routing strategies. This way, the results will be more general and also teach us about resilient network design.

Routing Algorithms. In the simulations, we tested two routing algorithms, which represent the classes of deterministic and load-balancing routing algorithms. The first, *fully-deterministic routing*, is suitable for latency-sensitive services, such as the web. It uses the Dijkstra algorithm [10] to find the path with the smallest number of hops. When many such paths exist between two endpoints, the same one is always chosen based on small random weights added to

Table 2. Network topologies in the simulations

Topology	Type	Edges	Nodes
Grid	regular mesh	yes	729 (27 × 27)
2D torus	regular mesh	no	729 (27 × 27)
3D torus	regular mesh	no	729 (9 × 9 × 9)
Hypercube	regular mesh	no	512 (9 dimensions)
Triangulated planar	irregular mesh	yes	729
Ad-hoc radio	irregular mesh	yes	729
Binary tree	tree-like	yes	511 (height 8)
Fat tree	tree-like	yes	720 (3 tiers, 24 pods)

the links. The second, *load-balancing routing*, is an example of an algorithm that takes network load into account in the routing. It also uses the Dijkstra algorithm but with the total number of active flows on the links in the considered path as the primary distance measure, and the hop count only as a secondary path selection criterion. This algorithm is *non-deterministic* in the sense that the route for a new flow depends on the other flows in the network and, thus, cannot be predicted reliably.

Network Topologies. We considered a broad range of network topologies with very different characteristics, which are shown in Table 2. The mesh topologies have been widely used for high-performance computing applications [7,12]. The triangulated planar topology, which is a Delaunay triangulation [8] of a set of points in a plane, and the ad-hoc radio topology, where geographically placed nodes communicate with those within the radio range, represent wireless mesh networks. The fat-tree topology is popular in data centers because of its high fault tolerance. Simple tree topology can also be deployed in small-scale networks. After experimenting with different node degrees, we chose binary trees for the simulations because they exhibit the properties of tree topologies at their most extreme. In the tree-like topologies, hosts were placed only in the leaf nodes. All the simulations treated the network as a single domain under a single controller, and thus we did not include inter-domain routing topologies.

Simulation Method. For each simulation, we picked one, two, or many compromised switches in random and measured the number of data flows that passed through them. This metric was chosen because it reflects how many flows are vulnerable to follow-up attacks (e.g. sniffing, man-in-the-middle) by the compromised switches. We first simulated the baseline case, i.e., with no topology poisoning. The baseline case was then compared with the different topology-poisoning scenarios. We created the fake links for each of the variant attacks described in Sect. 3.3 and repeated the experiment to see how many more data flows were captured by the compromised switches. Each such experiment was repeated 100 times with each simulation lasting 10 s with 50 000 data flows of at most 1 s each.

The attacks were implemented as follows. In the single-switch variant, we used the following simple heuristics for pairing up the neighbors for the fake links. On the rectangular networks, the fake links were formed between neighbors on the opposite sides of the compromised switch. In the fat tree topology, if the single compromised switch was in the aggregation layer, the fake links were created between the core and edge switches, and otherwise randomly. In the neighbor variant, the neighbors were always paired up randomly.

Most of the simulations were run with only one or two compromised switches because we wanted to learn the basic characteristics of topology poisoning attacks from these "atomic" cases. When the attacker has many compromised switches, the attack is composed of the atomic cases, but there is a combinatorial explosion in the possible configurations and parameters, such as choosing the fake links and relay nodes from among the compromised switches. For the same reason, we assumed a relay node to exist when assessing the impact of the attack variants and routing algorithms, and then measured in a separate experiment the availability of potential relay nodes (see Sect. 5.4).

5 Simulation Results

This section presents the evaluation results and the insights gained from them. We show the results for selected topologies that best illustrate the results.

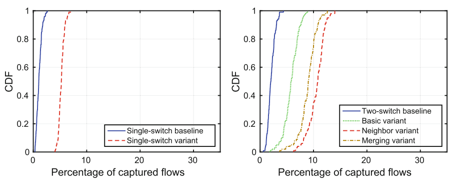
5.1 Topology Poisoning Compared to Baseline

Figure 1 illustrates how successful the attacker is in diverting traffic flows to the compromised switches. The CDFs show the percentage of captured flows. In each plot, the blue baseline (usually the leftmost line) shows how many data flows go through the compromised switches when they are passively sniffing network traffic. The further right the other lines are from the baseline, the more the attacker gains from creating the fake links. Our observations are as follows.

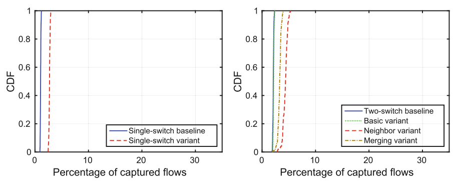
First, *topology poisoning increases significantly the attacker's ability to capture traffic flows*. That is, the compromised switches that mount the topology poisoning attacks are typically able to attract several times as many traffic flows as in the baseline case.

Second, *the neighbor and merging variants of the attack are clearly more successful than the basic variant*. This provides proof that our investigation into the attack variants was worth the trouble. The attacker can dynamically choose the most successful attack variant, and without considering the more powerful variant attacks, we might underestimate the threat of topology poisoning.

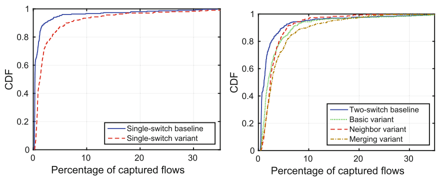
Third, *even the single-switch attack can divert significantly more flows than the one-switch baseline*. Since the single-switch attack can be mounted by each compromised switch locally and without coordination, it seems that the attacker would, at minimum, try this variant attack.



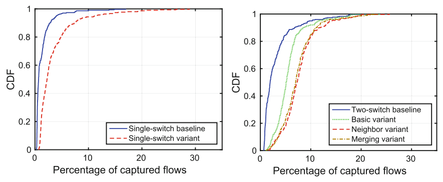
(a) Fully-deterministic routing, 3D torus



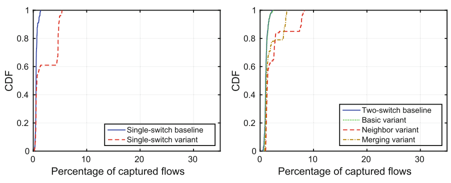
(b) Load-balancing routing, 3D torus



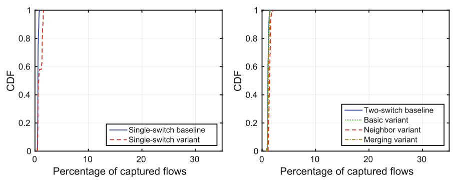
(c) Fully-deterministic routing, triangulated planar topology



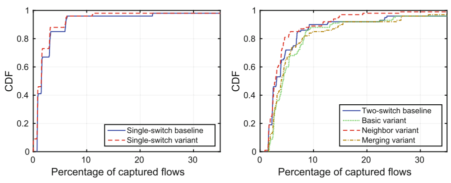
(d) Load-balancing routing, triangulated planar topology



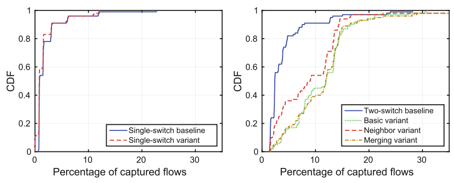
(e) Fully-deterministic routing, fat tree



(f) Load-balancing routing, fat tree



(g) Fully-deterministic routing, binary tree



(h) Load-balancing routing, binary tree

Fig. 1. CDF of the percentage of captured flows with different routing algorithms and network topologies (line further right means more captured flows)

5.2 Impact of Different Factors

From Fig. 1, we can see that there is big variation in the results between simulations. In some cases, the attacker is able to capture as much as 10–30% of all flows in the network, but usually less than 5–10%. Below, we analyze the factors that influence the attacker's success.

Load Balancing. One key observation is that in most cases, *the more load balancing or non-determinism there is in the routing, the less effective the topology poisoning attacks are.* This is illustrated by the comparison between the figures in the two leftmost columns to the two rightmost columns). The difference is especially prominent in the 3D torus and fat tree topologies.

The mitigating effect of load balancing can be explained by the fact that, if a fake link is able to capture many traffic flows, the controller soon thinks that the link is congested and routes the new flows around it. In general, any randomness or state-dependency in the routing means that the shortest path by hop count is not always used, and that reduces the attacker's ability to attract large numbers of flows to a small number of compromised switches. The effect is strongest in the 3D torus and fat tree because there are many paths of equal or almost-equal length, over which the network load can be distributed.

We also note that *the load-balancing is less effective in mitigating the neighbor and merging variant attacks.* This is because the attacks create multiple fake links, which have more apparent capacity than the single fake link in the basic variant.

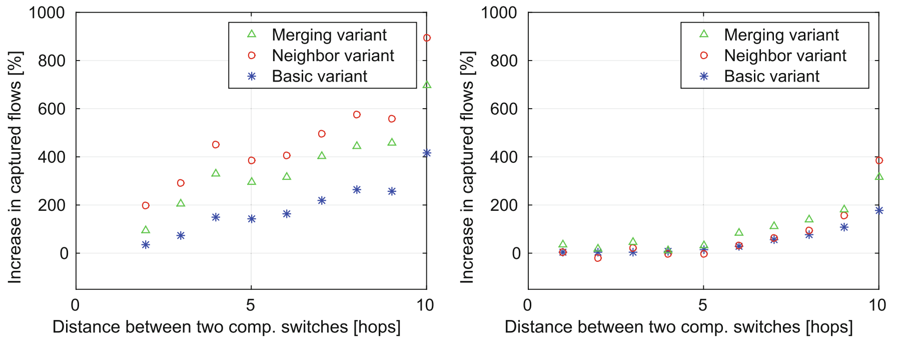
Network Topology. The fat tree topology has some properties that make it particularly resistant to topology spoofing (see Figs. 1e–1f). First, *the fat tree has so many alternative routes that the compromised switches capture few flows.* This is true both for the baseline and under the different attack variants. Second, the fat tree is so shallow that the length of the longest path in the tree is only 4. *It is unlikely that a randomly placed fake link would shorten the already very short routes in the fat tree.* The attack resistance of the fat tree comes at the cost of needing many redundant hardware links.

Figures 1g–1h of binary tree are striking because the neighbor variant attack is less successful than the basic variant and even the baseline. The result is explained by the fact that the pure tree topology is very sparse, i.e. has the minimum number of links to be connected. This kind of network becomes partitioned by the removal of a single switch or link. *The neighbor variant attack has tendency of breaking networks with sparse-graph topology, such as trees, and thus is not suitable for them.*

Finally, another key observation of this paper is that *an irregular or slightly randomized mesh network is more resistant to the single-switch variant attack than a mesh based on a simple regular pattern.* For example, adding some random diagonal links to a rectangular grid or randomizing the degree of switches in the triangulated planar graph weakened the single-switch attack. This tendency can be explained by the fact that a regular network structure has many equal-length shortest paths between two endpoints, and one small shortcut may be able to divert them all. The phenomenon is familiar from cities with a rectangular grid plan. If even just one of the rectangles is a park that can be crossed diagonally, that park attracts a large number of people walking through because it shortens so many paths. Graphs with irregular structure lack this effect, and in regular networks, it can be mitigated by adding randomness artificially.

Location of the Compromised Switches. It is obvious that central network locations are best for sniffing if the network has a center and an edge. For example, the grid, triangulated planar, and tree topologies have a center and edge, while the 2D torus, 3D torus, and hypercube do not.

For topology spoofing, the central network locations may not be as critical because the *fake links attract more traffic when their endpoints are more distant*



(a) Fully-deterministic routing, 3D torus (b) Fully-deterministic routing, triangulated planar topology

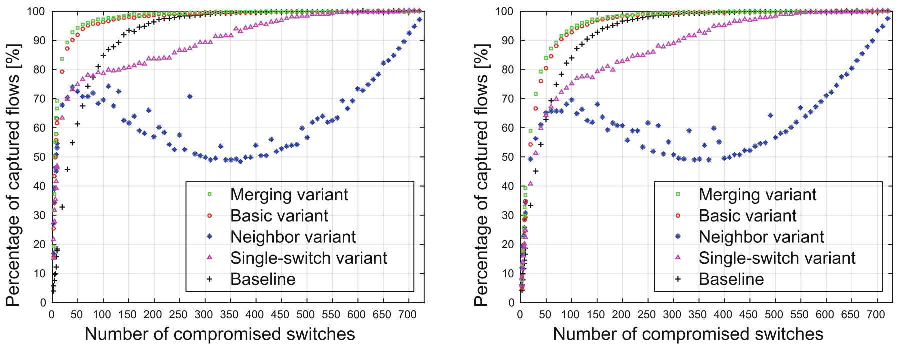
Fig. 2. Effect of the distance between compromised switches on the percentage increase of captured flows over the baseline

from each other. This is because, the longer jump a fake link makes, the bigger its effect on the apparent path lengths. Figure 2 depicts the relation between the distance of the two compromised switches and the average effect of the fake link on the number of captured flows (compared to the passive-sniffer baseline). The relation between the distance and the effectiveness of the fake tunnel is clear in networks with no center and edge, such as the 3D torus. In a network with a center, we could not come to any definite conclusion about the optimal location of the compromised nodes as it depends heavily on the network and on the traffic distribution.

5.3 Many Compromised Nodes

Finally, we get to discuss attacks with more than two compromised switches forming tunnels. Figure 3 shows the number of captured flows for the 2D torus when the attacker creates the maximum number of tunnels. The results for the other topologies are essentially similar.

The main lesson is that *a small fraction of compromised switches is sufficient to capture most network flows*. From the left-hand figure, we see that the some of the variant attacks, especially the neighbor variant, fail and perform worse than the baseline when the number of compromised switches grows above a certain threshold. This is due to the attacks breaking links and causing problems in routing. Naturally, a smart attacker will not create more tunnels than what gives optimal results. Overall, the insight for the attacker is that only a small number of compromised switches are needed, and if many are available, the main benefit is being able to choose good locations for the tunnel endpoints and to vary them over time.



(a) Fully-deterministic routing, 2D torus (b) Load-balancing routing, 2D torus

Fig. 3. Effect of the number of compromised switches on captured flows (the right-hand figure is a zoom to the important part of the left-hand figure)

5.4 Finding a Relay Node

The two-switch variant attacks are effective but they crucially depend on having a relay node through which traffic can be tunneled between the compromised switches. In the basic variant attack, a node r can act as the relay between two compromised switches c_1 and c_2 iff the real distance from c_1 to r is shorter than the distance via the fake link, and the same must hold in the other direction:

$$\begin{aligned} &d(c_1, r) < d(c_2, r) + 1 \text{ and } d(c_2, r) < d(c_1, r) + 1 \\ &\implies d(c_1, r) = d(c_2, r) \end{aligned}$$

That is, the relay node must be equidistant from the compromised nodes. For the neighbor and merging variants, there is no such simple formula because one specific neighbor of each compromised node needs to be reserved for communicating with the relay (and not be the endpoint of any fake link). In practice, the attacker can choose the tunnel endpoints simply by testing which tunnels work with the available relay nodes.

Simulations showed that the number of nodes that can act as the relay node varies remarkably. Their number tends to decrease slightly when the distance between the compromised nodes grows, but variations between individual simulations are far greater. Figure 4 illustrates these variations in the grid and triangulated planar topologies. The two red nodes represent the two compromised switches, while the green nodes represent the switches that can relay traffic between the red nodes. Overall, the average percentage of potential relays varied between 3–50 % of all switches (or hosts attached to them) depending on the topology and distance between the compromised nodes. Of course, the attacker may not have the luxury of choosing the location of the relay node, which usually is a compromised host or a host in the Internet. In that case, these probabilities give a rough indication of how often a fake link can be constructed with a given relay node.

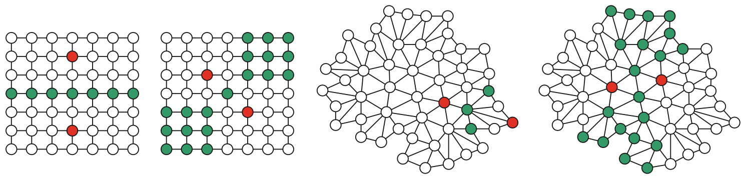


Fig. 4. Examples of possible relay nodes

When there are more than two compromised switches, it no longer makes sense to ask which are the potential relay nodes because the fake links influence each other's communication. Instead, fake links between the switches should be constructed one by one until the available relays cannot be used to create more.

6 Discussion

It is clear from the simulation results that topology poisoning attacks are attractive to an attacker who is looking for any way to expand a small foothold on a network. In the following, we will discuss the generality of the results and defensive strategies.

6.1 Generalizing the Results

The attacks were tested against OpenFlow, but the principles are more general. First, the link discovery with LLDP is not specific to OpenFlow. It and similar proprietary protocols are widely used for topology discovery. Second, our simulations were done on a high level of abstraction that does not depend on the exact control protocol or method of link discovery, so that the result can be generalized to most SDN technologies.

Regarding non-SDN networks, topology poisoning is not very interesting if compromised routers can achieve the same (and more) by tampering with the routing protocol. The topic of this paper becomes relevant when the attacker has no easier method of tampering with the routing than topology poisoning.

6.2 Defensive Strategies

Architectural and Routing Strategies. As observed in this paper, load balancing and non-determinism in the routing reduce the effects of topology poisoning because it becomes more difficult to divert large numbers of traffic flows to a small number of switches and links. Random route mutation [11], which is a technique where routes between endpoints are periodically randomized, is thus a very effective countermeasure against these attacks. Redundant links and paths, such as those in the fat tree topology, also mitigate the effects of the attacks.

Similarly, irregularity in the network topology reduces the impact of topology-poisoning attacks. In grid networks and other regular topologies that have many paths of (almost) equal length, small amounts of random variation, such as shortcut links, could be added artificially to counter the effect of topology poisoning.

Network like 2D and 3D torus, which have no edge, may be more resistant to attackers who are able to choose the location of the compromised nodes. This is because the network has no central locations that most data flows would traverse. This is definitely true for the baseline case, i.e. passive sniffing.

Static Detection Strategies. Any a-priory knowledge that the controller has about the network topology or the ports and links on the switches will naturally help in preventing topology spoofing. For example, the basic variant of the attack may require the attacker to create virtual ports on the switches, and the neighbor and single-switch variants may cause a usually busy switch to appear almost or entirely disconnected from the rest of the network (because it has become transparent). As a rule, however, it is not desirable to require manual configuration of such information about the network in the controller. This is because manual configuration is error-prone and not scalable in practice.

Dynamic Detection Strategies. The controller can test the links to detect fake ones. It can isolate each link in turn from the rest of the network, so that tunnel operation is prevented, and then forwarding packets though it with the OpenFlow commands. While such testing can be implemented entirely in controller software, it is difficult to do this without disrupt the normal operation of the network. Thus, this method is useful mainly during network bootstrapping or when new switches and (real or fake) links are added, but it is not practical for scanning a network that is already in operation. Load testing might detect that two fake links are not truly independent and, thus, may be based on the same tunnel, but that will also disrupt the normal network operation.

Another monitoring solution is to observe the latencies of links, such as the delays in LLDP or data flows. This might work because routing through fake links increases the latency in most cases. It may, however, be difficult to measure the forwarding time accurately, especially with in-band control.

Unfortunately, these dynamic detection techniques work poorly against the single-switch variant attack, which does not require any tunneling or create significant latency.

Hardware-Based Strategies. Without having access to the switch's private authentication key, the attacker is not able to spoof good switch behavior to the controller. Therefore, trusted computing technology, such as secure boot or storing the private authentication key on tamper-proof hardware, can make the attacks more difficult to implement and easier to detect. Trusted execution environments (TEE) built into the latest microprocessors also enable such protection of the switch's credentials.

7 Conclusion

A trusted central controller with a network-wide view is one of the major innovations in SDN and mitigates many of the threats against distributed routing algorithms. The view of the controller can, however, be poisoned in the link-discovery phase. Compromised switches can create fake links to the network and, thereby, divert traffic flows via themselves. This paper provides a classification and thorough analysis of topology poisoning attacks and the factors that influence their success. For example, we found that a single compromised switch may be able to mount a successful topology poisoning attack to increase its foothold in the networking. On the positive side, we found that irregularities in the network topology and load-balancing or nondeterminism in routing may mitigate the impact of these attack. The results, nevertheless, indicate that topology poisoning by a small number or rogue network elements is a serious concern in SDN if used to enable further attacks such as sniffing and man-in-the-middle.

References

1. IEEE standard for local and metropolitan area networks- station and media access control connectivity discovery. In: IEEE Std 802.1AB-2009 (Revision of IEEE Std 802.1AB-2005) (2009)
2. Alharbi, T., Portmann, M., Pakzad, F.: The (in) security of topology discovery in software defined networks. In: IEEE Conference on Local Computer Networks (LCN). IEEE (2015)
3. Antikainen, M., Aura, T., Särelä, M.: Spook in your network: attacking an SDN with a compromised openflow switch. In: Bernsmed, K., Fischer-Hübner, S. (eds.) NordSec 2014. LNCS, vol. 8788, pp. 229–244. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11599-3_14
4. Benton, K., Camp, L.J., Small, C.: OpenFlow vulnerability assessment. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM (2013)
5. Cao, J., et al.: The CrossPath attack: disrupting the SDN control channel via shared links. In: USENIX Security Symposium (2019)
6. Chi, P.W., Kuo, C.T., Guo, J.W., Lei, C.L.: How to detect a compromised SDN switch. In: IEEE Conference on Network Softwarization (NetSoft). IEEE (2015)
7. Choo, H., Yoo, S.M., Youn, H.Y.: Processor scheduling and allocation for 3D torus multicomputer systems. IEEE Transactions on Parallel and Distributed Systems (2000)
8. Delaunay, B.: Sur la sphere vide. Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk (1934)
9. Dhawan, M., Poddar, R., Mahajan, K., Mann, V.: Sphinx: detecting security attacks in software-defined networks. In: Network and Distributed System Security (NDSS) Symposium. USENIX (2015)
10. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**, 269–271 (1959)
11. Duan, Q., Al-Shaer, E., Jafarian, H.: Efficient random route mutation considering flow and network constraints. In: IEEE Conference on Communications and Network Security (CNS). IEEE (2013)
12. Duncan, R.: A survey of parallel computer architectures. Computer **23**, 5–16 (1990)
13. Guha, A., Reitblatt, M., Foster, N.: Machine-verified network controllers. In: ACM SIGPLAN Notices. ACM (2013)

14. Hong, S., Xu, L., Wang, H., Gu, G.: Poisoning network visibility in software-defined networks: new attacks and countermeasures. In: Network and Distributed System Security (NDSS) Symposium. USENIX (2015)
15. Huston, G., Rossi, M., Armitage, G.: Securing BGP – a literature survey. Communications Surveys Tutorials, IEEE (2011)
16. Jero, S., Koch, W., Skowyra, R., Okhravi, H., Nita-Rotaru, C., Bigelow, D.: Identifier binding attacks and defenses in software-defined networks. In: USENIX Security Symposium (2017)
17. Kazemian, P., Chan, M., Zeng, H., Varghese, G., McKeown, N., Whyte, S.: Real time network policy checking using header space analysis. In: NSDI. USENIX (2013)
18. Khurshid, A., Zhou, W., Caesar, M., Godfrey, P.: Veriflow: verifying network-wide invariants in real time. ACM SIGCOMM Comput. Commun. Rev. **13**, 15–27 (2012)
19. Koponen, T., et al.: Onix: A distributed control platform for large-scale production networks. In: OSDI. USENIX (2010)
20. Kreutz, D., Ramos, F., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. In: Proceedings of the IEEE (2015)
21. Kreutz, D., Ramos, F., Verissimo, P.: Towards secure and dependable software-defined networks. In: ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM (2013)
22. McKeown, N., et al.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**, 69–74 (2008)
23. Mizrak, A.T., Cheng, Y.C., Marzullo, K., Savage, S.: Fatih: detecting and isolating malicious routers. In: IEEE International Conference on Dependable Systems and Networks. IEEE (2005)
24. Papadimitratos, P., Haas, Z.: Secure link state routing for mobile ad hoc networks. In: Symposium on Applications and the Internet Workshops (2003)
25. Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., Gu, G.: A security enforcement kernel for OpenFlow networks. In: ACM Workshop on Hot Topics in Software Defined Networks. ACM (2012)
26. Porras, P., Cheung, S., Fong, M., Skinner, K., Yegneswaran, V.: Securing the software-defined network control layer. In: Network and Distributed System Security (NDSS) Symposium. USENIX (2015)
27. Röpke, C., Holz, T.: SDN rootkits: subverting network operating systems of software-defined networks. In: Bos, H., Monrose, F., Blanc, G. (eds.) RAID 2015. LNCS, vol. 9404, pp. 339–356. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26362-5_16
28. Shin, S., Porras, P.A., Yegneswaran, V., Fong, M.W., Gu, G., Tyson, M.: FRESCO: modular composable security services for software-defined networks. In: Network and Distributed System Security (NDSS) Symposium. USENIX (2013)
29. Wang, F., Vetter, B., Wu, S.F.: Secure routing protocols: Theory and practice. North Carolina State University, Technical report (1997)
30. Wen, X., et al.: SDNShield: reconciliating configurable application permissions for SDN app markets. In: IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE (2016)

Client-Side Vulnerabilities in Commercial VPNs

Thanh Bui(✉), Siddharth Rao, Markku Antikainen, and Tuomas Aura

Aalto University, Espoo, Finland
thanh.bui@aalto.fi

Abstract. Internet users increasingly rely on commercial virtual private network (VPN) services to protect their security and privacy. The VPN services route the client's traffic over an encrypted tunnel to a VPN gateway in the cloud. Thus, they hide the client's real IP address from online services, and they also shield the user's connections from perceived threats in the access networks. In this paper, we study the security of such commercial VPN services. The focus is on how the client applications set up VPN tunnels, and how the service providers instruct users to configure generic client software. We analyze common VPN protocols and implementations on Windows, macOS and Ubuntu. We find that the VPN clients have various configuration flaws, which an attacker can exploit to strip off traffic encryption or to bypass authentication of the VPN gateway. In some cases, the attacker can also steal the VPN user's username and password. We suggest ways to mitigate each of the discovered vulnerabilities.

1 Introduction

Virtual private networks (VPNs) [33] were originally developed for connecting geographically distributed corporate networks to each other and also for connecting remote workers to the employer's intranet. However, one of the most common uses of VPNs today is to protect regular Internet users who seek improved security and privacy. Such users perceive a need for a VPN in many situations, such as when accessing the Internet over public Wi-Fi (e.g. at a cafe or airport) or to hide their online activities from an oppressive entity (e.g. government). Because of the increased demand, a large number of *commercial VPN services* have appeared in the market.

Commercial VPNs typically function by tunneling the user's Internet traffic through a trusted remote server before it is forwarded to its final destination. This achieves two goals: (1) the traffic is protected by an encrypted VPN tunnel and (2) the destination server does not learn the real IP address of the client. The commercial VPN providers usually provide native client applications with graphical user interfaces, which automatically set up the VPN connection for the users. For more technically savvy users who prefer not to install the provided applications, they usually give instructions for configuring the built-in VPN client in the user's operating system (OS) to work with their servers.

© Springer Nature Switzerland AG 2019
A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 103–119, 2019.
https://doi.org/10.1007/978-3-030-35055-0_7

While the commercial VPN services have undergone severe scrutiny, only a few of them have taken a closer look at the client configuration [12,30]. They revealed misconfiguration issues that lead to user de-anonymization and traffic leakage. Our work extends this theme with the focus on the security of the VPN tunnels, namely whether they are encrypted and authenticated properly.

Motivation. Our work was primarily motivated by the observation that many commercial VPN providers configure L2TP/IPsec, a popular VPN protocol, in an insecure way. Specifically, the protocol relies on IPsec [13] to provide the secure transport, but many VPN providers use one pre-shared key for all users to authenticate the IPsec tunnels, which is usually publicly available online or can be discovered by examining the client configuration. If an attacker knows the pre-shared key, it can perform a man-in-the-middle (MitM) attack on the VPN connection and obtain all the network traffic to and from the victim's computer. This problem was discussed on public forums in 2016 [3]. However, when we re-analyzed the 14 insecure commercial VPN services mentioned in the discussion, we found that 10 of them were still using the insecure configuration. Since this security issue remains opaque to most end-users, we feel that it is important to systematically scrutinize the client configurations of commercial VPN services for flaws that could undermine the user's security and privacy.

Contributions. In this paper, we study how popular commercial VPN providers set up, or how they instruct users to set up, desktop VPN clients for common VPN protocols. Our study covers three common desktop OSs: Windows, macOS and Ubuntu. The study reveals various vulnerabilities in the configurations of VPN clients, which allow attackers to strip off traffic encryption or to bypass server authentication. By exploiting these vulnerabilities, attackers can intercept network traffic to and from the victim. Some of the vulnerabilities also allow the attacker to steal user credentials for authenticating to the VPN gateway. While each of the vulnerabilities alone might seem like a trivial mistake, together they indicate a serious lack of security-awareness across the commercial VPN industry, and we feel that it is the responsibility of the research community to raise the issue. We also provide guidelines on fixing the vulnerabilities. Through this work, we hope to raise awareness among the commercial VPN providers about common configuration mistakes and how they can be avoided.

2 Commercial VPN Services

Commercial VPNs (also known as personal or consumer VPNs) are subscription-based services available to regular Internet users. They are typically used for personal purposes, such as accessing geoblocked media contents, securing online activities while on public Wi-Fi networks, and avoiding censorship and surveillance by local governments and access-network operators.

Most commercial VPN providers have a *native client application*, which sets up the VPN connection for the user. These applications are usually available for Windows and macOS users. To use the application, users must first sign in with their VPN credentials. The application then pulls configuration data, such as VPN server addresses, roots of trust for the authentication, and VPN-client credentials (which are not necessarily the same as the user-login credentials), from the VPN provider's server and configures the VPN tunnel for later use. It typically allows the user to choose from many different VPN protocols for the tunnel implementation.

The client applications are usually available only for Windows and macOS users. The applications rely on the OS's built-in VPN client functionality whenever it exists. Windows is bundled with implementations of the PPTP, L2TP/IPsec, SSTP and IKEv2 protocols, while macOS comes with L2TP/IPsec, IKEv2 and Cisco IPsec client functionality. For the protocols that have no built-in support in the OS, the commercial VPN providers include *third-party client binaries* in their native applications.

For the users of the OSs that the VPN providers do not have native client applications or for those who prefer not to install the provided applications, the VPN providers give instructions on their websites for configuring the OS's built-in VPN client. They may also give advices on installing and configuring third-party clients to use with their service.

3 Study of Commercial VPN Services

3.1 Adversary Model

The object of our study is the way commercial VPN services make use of the common VPN protocols and tunnels. We consider two types of attackers whose ultimate goal is to bypass the protection mechanisms of the VPN connection and steal sensitive data sent or received through it: *network attacker* and *local attacker*.

Network Attacker. We consider an active network attacker who can intercept and modify network traffic originating from and destined to the user's machine. The attacker could, for example, be a rogue hotspot operator at a hotel or airport, or a compromised core-network operator. This is the standard attacker model for network security.

Local Attacker. The VPN client software on the user's computer often comprises multiple components that are communicated with each other via inter-process communication (IPC) (e.g. network sockets, files on disk). For example, the GUI component may use an IPC channel to sent the VPN configuration to a third-party client binary. If such channels are not protected properly, *non-privileged* processes of other users [14] could exploit them to steal sensitive information or to modify the VPN connection settings. We included this type of attackers to the study because the vulnerability of VPN clients to it is currently not well understood and the attacks are different from those on the network.

3.2 Methodology

We selected 30 commercial VPN services based on popularity and advertised features (refer to Table 1). As a rough estimate of popularity, we searched for "best VPN services" on Google and counted how many times each service was mentioned in the resulting pages. The idea was to identify the services that normal users would be most likely to choose. Among the popular services, we prioritized those that support a higher number of VPN protocols. We focused on standard VPN protocols, including *PPTP*, *L2TP/IPsec*, *IKEv2*, *Cisco IPsec*, *SSTP*, *OpenVPN*, and *SoftEther VPN*.

We analyzed the selected VPN services with a two-step process as follows.

Configuration Analysis. In each of the commercial VPN client applications, we first looked at the way the application creates and configures the VPN connection. When the VPN service providers recommended a built-in client in the OS or a third-party client, we scrutinized the provided configuration instructions and unchanged default settings. In both cases, we looked for potential misconfigurations and architectural mistakes that might compromise the security of the resulting VPN connection. We did not try to find flaws in the cryptographic protocols themselves or code-level implementation errors.

Experimental Verification. When we found a potential client-configuration issue, depending on the type of the attack, we verified it as follows.

For network attacks, we first created a fake VPN server to intercept connections from the client to the gateway server. We then routed the VPN client's traffic to the fake server as follows. When testing a commercial VPN client application, we edited `/etc/hosts` to map the true VPN server's domain name to the fake server's IP address. When testing the instructions for configuring a built-in or third-party VPN client, we simply followed the instruction but gave the fake server's IP address as the gateway address. These methods sufficiently emulate the behavior of a network attacker that intercepts the connections on an untrusted access or core network. Finally, if the VPN client successfully connected to the fake server without dropping the connection or alerting the user, we concluded that the client is vulnerable to the attack currently under test.

For local attacks, we created two user accounts on a test machine: one acted as the honest user and the other as the attacker. The attacker here is a standard user with no administrative privileges (a guest account can be equally used). We wrote a script to exploit the potential vulnerability in the IPC, executed it in the attacker's login session, and checked whether it succeeded in exploiting the vulnerability in the VPN client application that was running in the honest user's login session.

4 Study Results

This section describes the vulnerabilities that we found in the client configuration of the selected commercial VPN services. Table 1 summarizes our findings.

4.1 Point-to-Point Tunneling Protocol

Point-to-Point Tunneling Protocol (PPTP) [18] was created by Microsoft, and it is one of the oldest VPN protocols. It has well-known weaknesses [19,22,26,31]

Table 1. Discovered vulnerabilities (✗: vulnerable, ✓: not vulnerable, –: not applicable, W: Windows, M: macOS, U: Ubuntu, N: Network attacker, L: Local attacker)

	PPTP	SSTP	IKEv2	OpenVPN	SoftEther	Cisco IPsec	Fallback
OS	W	U	U	W	W,M	W,M,U	W,M
Attacker type	N	N	N	L	N, L	N	N
Astrill	✓	–	–	✓	–	✗	–
BoxPN	✗	–	–	✗	–	✗	–
CactusVPN	✗	–	–	✓	✗	–	–
CyberGhost	✓	–	–	✓	–	✗	✓
ExpressVPN	–	–	–	✗	–	–	✗
FastestVPN	✓	–	–	✗	–	✗	–
FrootVPN	✗	–	–	✗	–	✗	–
GooseVPN	–	–	–	✗	–	✗	✗
Hide.me	✓	–	✗	✓	✗	–	✓
HideMyAss	✗	–	–	✗	–	✗	–
ibVPN	✗	–	–	✓	✗	✗	✓
IPVanish	✗	–	–	✗	–	✗	–
IVPN	–	–	✓	✓	–	–	–
LimeVPN	✗	–	–	–	✗	–	–
NordVPN	–	–	✗	✓	–	–	–
OverplayVPN	✓	–	–	✓	–	–	–
Perfect-Privacy	–	–	–	✓	–	✗	✓
PersonalVPN	✗	–	✓	–	–	✗	–
PIA	✓	–	–	✓	–	–	–
PrivateVPN	✓	–	–	✓	–	✗	–
ProXPN	–	–	–	✓	–	–	✗
PureVPN	✗	✗	–	✗	✗	–	✓
RocketVPN	–	–	–	–	✗	–	–
SaferVPN	✗	–	–	✓	–	–	✗
StrongVPN	✗	✗	✗	✗	–	✗	–
TorGuard	✗	✗	–	✓	–	✗	–
Trust.Zone	–	–	–	–	✗	–	–
UnblockVPN	✗	✗	–	–	–	–	–
VyprVPN	✗	–	–	✓	–	–	–
24VC	–	–	–	–	✗	–	–

and is no longer considered secure. Nevertheless, the protocol remains widely deployed and used because many firewalls do not block it. Our goal is to see whether it has additional weaknesses that could further compromise the users' security and privacy.

To establish a PPTP connection, an IP tunnel is first created between the client and the server, and then a Point-to-Point Protocol (PPP) [32] session is instantiated inside the tunnel. The PPP session goes through three phases:

1. *Authentication:* The communication endpoints first authenticate each other. Most commercial VPN services implement MS-CHAPv2 as the only authentication method.
2. *Negotiation:* If the authentication is successful, the client and the server negotiate parameters such as the encryption scheme and the DNS servers. The only encryption scheme that PPP supports is Microsoft Point-to-Point Encryption (MPPE) [28]. A well-configured PPTP server usually enforces MPPE with 128-bit keys on its connections. The encryption key is derived from the authentication in the previous phase. It is important to note that all packets until completion of this phase are transmitted in plain text.
3. *Data exchange:* Finally, the client starts communicating network traffic with the server by encrypting it as per the negotiated encryption scheme.

Optional Encryption. Windows by default does not enforce encryption on any VPN connection. We found that many of the selected commercial VPN services do not instruct their users to change this setting while configuring PPTP with the built-in client on Windows. A network attacker can take advantage of this behavior to perform server impersonation as follows. First, the attacker acts as a man-in-the-middle to forward traffic between the client and the honest server until the authentication phase is finished. The attacker then switches to performing server impersonation and negotiates with the client not to encrypt the data exchange. The client agrees to this because it is not mandatory to use encryption. As the result, the attacker obtains all traffic of the victim just as if no VPN was used.

4.2 SSTP

Secure socket tunneling protocol (SSTP) [25] is another VPN protocol created by Microsoft. It also utilizes PPP to transport network traffic, but it encapsulates the PPP packets in HTTPS. An SSTP connection is established as follows:

1. The client opens an HTTPS connection to the server. It authenticates the server by verifying the server's TLS certificate as in any HTTPS connection.
2. If the TLS authentication succeeds, the client begins SSTP negotiation by sending a `Connect-Request` message to the server. The server replies with a `Connect-Acknowledgement` message that contains a nonce to be used later.
3. Both sides perform the PPP authentication, deriving a session key for MPPE. Like in PPTP, MS-CHAPv2 is usually used for this. However, in this case, it is protected against active and passive attacks by the HTTPS encryption.

4. The client sends a `Call-Connected` message that contains the nonce received from the server, a hash of the server certificate from the HTTPS handshake, and a message authentication code (MAC) that is computed over the message with the MPPE key derived during the PPP authentication. This message cryptographically binds the PPP session to the server identity from the outer TLS authentication.
5. The endpoints perform the PPP negotiation and then start to exchange network traffic. SSTP does not use MPPE encryption. Instead, it relies entirely on HTTPS for the secure delivery of its messages.

Windows has a built-in SSTP client. On macOS and Ubuntu, VPN services usually suggest the user to install `EasySSTP` [1] and `sstp-client` [7], respectively. `EasySSTP`, however, has been obsolete since 2013, and it no longer works on the latest version of macOS (10.14) at the time of writing this paper.

Ignored Certificate Verification Failures. We tested the SSTP client on Windows 10 and the latest `sstp-client` (v1.0.11) to see whether they perform the mutual authentication properly. While the Windows client does this correctly, `sstp-client` does not consider whether the server's certificate is trusted. By inspecting the source code of `sstp-client`, we found the reason for this unexpected behavior: `sstp-client` is integrated into Ubuntu network connection manager, which allows the user to configure whether the connection should be terminated when the certificate verification fails, but it ignores certificate verification errors regardless of this setting.

Ignoring the certificate verification failure allows the network attacker to perform server impersonation attacks as follows. First, the fake server presents a self-signed TLS certificate to the honest client. It then connects to the honest server pretending to be the client. The attacker forwards traffic between the honest client and the honest server until the PPP authentication is completed and the attacker sees the `Call-Connected` message (Step 4 above). The attacker then stops forwarding traffic to the honest server and finishes the PPP negotiation by itself. When the SSTP connection is successfully established, the fake server can act as the VPN gateway and obtain all the victim's traffic.

4.3 IKEv2

IKEv2 is a more modern VPN protocol based on IPsec. It uses IKEv2 [21] for authentication as well as establishing and maintaining security associations. One improvement of IKEv2 over IKEv1 is that the new protocol allows each endpoint to use a different authentication method. IKEv2 also supports EAP [11], extending the selection of available authentication methods.

In a typical IKEv2 connection that is set up by VPN services, the server authenticates itself to the client with a certificate while the client authenticates to the server with EAP-MSCHAPv2 [20], which is basically MS-CHAPv2 encapsulated in the EAP protocol. The connection is established as follows:

1. *Initial exchange:* The client and server negotiate security parameters, such as cryptographic algorithms, and exchange nonces and Diffie-Hellman (DH) values. After that, each party computes the shared session keys, which will be used for protecting all the following messages. These values will also be used for constructing the first security association (SA).
2. *Server authentication:* The server authenticates itself to the client with its certificate (or certificate chain) and a signature on the SA data.
3. *Client authentication with EAP:* The client is then authenticated with the EAP-MSCHAPv2 protocol. After completion of the protocol, the client and server exchange MACs to bind the EAP authentication to the created SA. The MACs are calculated over the SA data with the session key produced by the MS-CHAPv2 protocol.

Both Windows and macOS have built-in support for IKEv2. On Linux systems such as Ubuntu, commercial VPN services usually instruct their users to install StrongSwan [8], an open-source IPsec implementation, for the client.

Unspecified Server Name. To establish an IKEv2 connection with StrongSwan, the user has to create a profile for the connection. Several commercial VPN providers in our study instruct their users to create the profile as follows (only important parts are shown).

```
leftauth = eap-mschapv2
...
right = <server-address>
rightauth = pubkey
rightid = %any
```

Left and right indicate the client and the server, respectively. With such a profile, the server uses a public key for authentication while the client uses EAP-MSCHAPv2. The problem is with the `rightid` setting, which tells how the server should be identified in the authentication. Since it is set to `%any`, the client will accept any certified server regardless of its identity. Consequently, the network attacker can pick any domain that it owns and purchase a certificate from a widely trusted CA. It can then impersonate the server in the server authentication step because the client does not check the name in the certificate.

Fortunately, MS-CHAPv2 actually provides mutual authentication with the user password. The binding of this authentication to the SA prevents the attacker from completing the protocol without knowing the password. Thus, the misconfiguration effectively reduces the security of IKEv2 to that of MS-CHAPv2, which is significantly weaker [22,31].

4.4 OpenVPN

OpenVPN [4] appears to be the most widely supported protocol by commercial VPN services. It uses TLS as the underlying authentication and key exchange protocol. Commercial VPN services deploy OpenVPN in the client-server mode, in which the server authenticates itself to the client with an X.509 certificate signed by a CA that the client trusts while the client proves its identity with a username and password.

Despite the wide range of configuration options that OpenVPN supports, we did not find any broken configuration examples that would allow the network attacker to compromise the OpenVPN connection. We found, however, that the local attacker, i.e. non-privileged local users and processes (see Sect. 3.1), can steal the username and password that are used for authenticating the client.

To support OpenVPN, commercial VPN services include the open-source `openvpn` client binary in their client software and run it as a daemon. To create an OpenVPN connection, the `openvpn` daemon needs configuration information including the server address, server name, and trusted CA certificate. There are two ways of delivering the configuration information to the daemon: the client GUI application can either write the configuration to a file or pass it to the daemon as command-line parameters. Additionally, the daemon will need the client's VPN username and password, which can be specified either in the configuration file or via the management interface described below.

The OpenVPN daemon supports a management interface [5], which allows administrative control via a TCP connection. By default, it only accepts connections on the `localhost` interface. The advantage of the management interface is that the user can avoid saving the client credentials into the configuration file on the disk. The commercial VPN application first starts the `openvpn` daemon with all the necessary configuration options, except the client credentials, and puts it on hold with the `management-hold` option. The application then connects to the management interface of the daemon, gives it the username and password, and finally releases the connection from the hold state.

Credential Leakage. Some commercial VPN client applications are careless when passing the username and password to the OpenVPN daemon. Specifically, the VPN applications store the credentials in configuration files that are readable to all users on the client computer. Therefore, the local attacker can capture this sensitive information. Some of these services remove the credentials from the file after the connection has been established, but this still leaves a window of a few seconds to capture the information.

4.5 SoftEther VPN

SoftEther VPN [6] is another VPN protocol with an open-source implementation that tunnels Ethernet frames over HTTPS. Similar to OpenVPN in the client-server mode, the SoftEther VPN server proves its identity to the client with a TLS certificate while the client has a username and password for its authentication.

The SoftEther VPN connection establishment is implemented with two binaries: `vpncmd`, the command-line administrative tool, and `vpnclient`, its command execution worker. The `vpnclient` worker process runs a TCP server on port 5555 for receiving administrative commands. By default, the TCP server accepts connections only from the `localhost` interface. It can be configured to require password authentication.

By issuing commands to the `vpnclient` TCP server, `vpncmd` can perform administrative operations such as creating a new VPN profile or editing an existing one, or starting and stopping a VPN connection with an existing profile. SoftEther VPN profiles have various configurable options, including whether the client should check the server's TLS certificate or not.

No Server Verification. Hide.me supports SoftEther VPN on its GUI. However, the `CheckServerCert` parameter is set to `false` in its client configuration. Consequently, the client does not verify the server's certificate. This allows the network attacker to perform a MitM attack on its connections and obtain the victim's credentials and network traffic.

Some other commercial VPN services support SoftEther VPN by providing instructions on how to connect to their servers with the SoftEther VPN GUI application [6]. When creating a new SoftEther VPN connection, the GUI allows the user to choose whether the client verifies the server's certificate. The default setting is `False`, and the VPN services do not tell their users to change the setting. Thus, they are vulnerable to the same attacks as Hide.me.

Wrong VPN Server. We found another problem with Hide.me, which is that its GUI does not require password authentication on the management interface of the `vpnclient` process. This allows the local attacker to connect to the interface and, for example, launch a new VPN connection that routes all network traffic of the victim to a malicious server under the attacker's control.

4.6 Cisco IPsec

Cisco IPsec [15] is widely used in enterprise VPNs. However, it is also supported by several commercial VPN services. Like L2TP/IPsec, the protocol uses IPsec to tunnel traffic. The main distinguishing feature of Cisco IPsec is that, after the communicating nodes have completed the conventional IKEv1 authentication, an additional phase of Extended Authentication (XAUTH) [29] is performed to authenticate the user. XAUTH allows various types of user authentication, such as challenge-response and one-time password.

Known Pre-shared Keys. We found that all of the commercial VPN services that support Cisco IPsec in our study use a pre-shared key to authenticate the IPsec tunnel. Learning from the experience of L2TP/IPsec, it is rather obvious to ask where the endpoints get the pre-shared key. The user interfaces of the commercial VPN clients do not have a way of entering such a key. Perhaps unsurprisingly, we found that the commercial VPN services have fixed pre-shared keys also for Cisco IPsec. This allows the network attacker to perform MitM attacks on these IPsec connections and to obtain all the network traffic.

4.7 Fallback Strategy

As mentioned in Sect. 2, commercial VPN services usually have a list of protocols from which the user can choose on the GUI of the provided application.

A special option that several services in our study support is *Automatic*, which means that the application will automatically select the protocol for the user. In particular, the application tries different protocols one after another until it succeeds in creating a VPN connection. Users typically choose this option if the firewall in the access network blocks some VPN protocols, or if they do not want to understand the technical intricacies of choosing the right protocol.

Fallback to Weak Option. It is easy to see that the security of the fallback strategy is equal to that of the weakest option which the application is willing to try. The network attacker can simply block all other connection attempts. We found that some commercial VPN services include L2TP/IPsec with publicly-known pre-shared key as an option in their automatic mode. For example, ExpressVPN attempts to use the following protocols: OpenVPN, SSTP, and L2TP/IPsec. If the network attacker blocks the first two (e.g. by filtering the corresponding ports), it effectively forces the client to use L2TP/IPsec, and thus, the attacker can perform a MitM attack on the VPN connection and obtain the traffic.

Windows also provides similar fallback strategies when RRAS [23] is used to create the VPN connection. With the `VpnStrategy` option [24], RRAS allows the developers to choose the order in which it attempts VPN protocols until the connection is successfully established. ProXPN, when configuring its IKEv2 client on Windows, instead of setting `VpnStrategy` to 5 (i.e. attempting IKEv2 only), sets the option to 8, which effectively tells the client to try the following protocols in order: IKEv2, SSTP, PPTP, and L2TP/IPsec. This causes the application to suffer from the same vulnerability in the automatic mode as ExpressVPN.

5 Mitigation Solutions

In this section, we discuss potential solutions to the issues presented in Sect. 4.

PPTP. As explained in Sect. 4.1, PPTP encryption is optional in the Windows implementation. Fixing this issue is straightforward: the VPN service providers simply need to tell Windows users to change the *Data encryption* setting of the PPTP connection adapter from *Optional encryption* to *Maximum strength encryption*. This enforces MPPE encryption with a 128-bit key on the connection. A more sustainable solution would be for Windows to employ strong encryption by default. While such changes to default settings are not always feasible for backward compatibility reasons, *secure by default* is one of the key principles in designing secure systems.

SSTP. The `sstp-client` in Ubuntu ignores certificate verification failures. As pointed out in Sect. 4.2, the flaw is in the `sstp-client` library code. Until it is fixed, commercial VPN services should explicitly instruct their users to not use `sstp-client` and possibly provide an alternative. The broader issue here is that modern software development practices create complex dependencies on free and third-party components, for which there may not be guaranteed maintenance.

One would expect security-critical services, such as commercial VPNs, to manage their dependencies carefully.

IKEv2. Setting the `rightid` value of a StrongSwan's IKEv2 configuration to `%any` could be useful, e.g. for testing a VPN server but not for production purposes. To fix the problem, the commercial VPN providers should give clear instructions to the users to set `rightid` to the server's domain name or to the Distinguished Name from the server's certificate [2].

The security failure here arises from the fact that it is easier to get the service to work with insecure wildcard settings than to find out and configure the correct server name. Even if the VPN service providers documented the correct usage, some users would find easier, insecure configuration entries online. Probably the only safe solution in this case is to automate the setup process and to audit the configuration regularly for unsafe changes by the user.

OpenVPN. The obvious solution to unauthorized users reading the OpenVPN configuration file is to set the access controls on the file, which is world-readable by default. Another solution is to avoid writing the VPN username and password to the file by communicating them over the management interface.

Storing secrets on the local machine is a problem encountered commonly by security-critical software. There is no perfect solution but making use of OS's access control is a good starting point. The next step might be a secure hardware module for storing client credentials.

SoftEther VPN. The management interface of SoftEther VPN must be protected with proper password authentication because without authentication, it allows any local user or process to control the VPN connections. The higher-level issue here is that TCP connections to `localhost` are not inherently secure and may require application-level authentication between the client and server [14].

To fix the server verification problem that we described, the `CheckServerCert` parameter of the SoftEther VPN connection must be set to `true` so that the client verifies the server's certificate during the establishment of the connection. This can be done either by changing the default value of `CheckServerCert` in SoftEther, or the VPN services can provide explicit instructions to their customers to do the same. This again highlights the importance of safe default values and the danger of allowing easy but insecure settings.

L2TP/IPsec and Cisco IPsec. The commercial VPN services that support the L2TP/IPsec and Cisco IPsec protocols share the same problem of using known pre-shared key for the IPsec authentication. Before considering solutions, it is worth discussing the reason for the use of fixed keys in these IPsec-based VPN protocols. They both use IKEv1 in the Main Mode, in which the server selects the pre-shared key by the IP address of the client. IKEv1 and the entire IPsec architecture reside in the IP layer, and thus the IP address is the only clue available for them about the client identity. Since the clients of the commercial VPN services practically always have dynamic IP addresses, it is not possible for the VPN gateway to support client-specific pre-shared keys. There have been proprietary proposals for sending a hint about the client identity to the server, but the rather historical IKEv1 protocol and its implementations have not been updated to support such new features in an interoperable way. Thus, the commercial VPN services have fallen back to the insecure practice of sharing the same key between all clients.

A solution is to switch from pre-shared keys to certificate authentication. For this, the commercial VPN services must obtain certificates for their VPN servers from a widely trusted commercial CA. The client certificates, on the other hand, can be provisioned by the VPN service provider itself. Client-side authentication in IKEv1 is not extremely critical anyway because the client user is authenticated separately with username and password in the later phases of the VPN protocols. It would, nevertheless, be a good practice to authenticate both the client device (with a certificate) and the client user (with username and password), so that devices and users can be revoked individually.

Fallback Strategy. Since the security of a fallback strategy is equal to the weakest allowed option, L2TP/IPsec and PPTP with their known weaknesses should be disabled in any automatic protocol selection process.

As mentioned earlier, one of the main reasons why commercial VPN services provide fallback options is to bypass firewall filters. The fallback strategy for firewall traversal gives an advantage to any malicious access-network operator or oppressive government that wants to attack VPN users. An alternative approach would be to only try safe firewall traversal techniques such as using server ports that are usually not blocked (e.g. 443).

When using RRAS to create VPN connections on Windows clients, the `VpnStrategy` setting should never be set to 0, 2, 4, 6 or 8 because all these values instruct Windows to attempt IKEv2, SSTP, PPTP, and L2TP/IPsec, just in different orders, until one succeeds. In an ideal world, Windows would stop supporting protocols and configuration options with known vulnerabilities.

6 Responsible Disclosure

We have reported all the vulnerabilities that we discussed in this paper to the corresponding VPN service providers. At the moment of writing, 15 of the tested providers have responded to us. They acknowledged all the problems and have fixed all of them, except the pre-shared key issue in L2TP/IPsec and Cisco IPsec, which none of them agrees to fix. They argued that the protocols are not secure and while they support them, they do not encourage their users to use them.

In addition, we reported the optional encryption problem of PPTP to the Microsoft security response team. While they acknowledged the problem, they are already looking into deprecating PPTP, and thus no immediate fixes will be released. Similarly, SoftEther team acknowledge the problem about the default value of the `CheckServerCert` parameter. However, they hesitate to change the default value as they believe it is the responsibility of the commercial VPN providers. We have reported the certificate verification problem to the author `sstp-client` but have not received a response.

7 Discussion

It appears that commercial VPN services compete by providing the maximum number of features, such as different VPN protocols, and maximum ease of use, and that security is a secondary concern for them. For example, the issue of using publicly known pre-shared keys for L2TP/IPsec has been known for years, but it has not been addressed by most VPN services. While this issue could be solved by provisioning certificates to the clients as discussed in Sect. 5, that would be an administrative hurdle for the VPN service providers and might scare away non-expert customers. This could be the reason why they have opted to continue the insecure services rather than endanger their business growth.

There are also many entirely unnecessary flaws in the VPN client settings, such as not checking the server certificate or server name. These appear to indicate lack of technical knowledge or security awareness by the service providers. We hope that the current paper will, at least to some extent, increase awareness about the importance of correct VPN configuration among the commercial VPN developers and operators.

We further observed the importance of developer documentation and configuration examples for third-party VPN software components that are used as building blocks in the commercial VPN services. Let us take OpenVPN for a positive example. It has detailed documentation of all the software configuration options as well as best-practice guidelines for building secure systems. Also, the OpenVPN software warns the user about insecure settings. These are probably the reasons why none of the commercial VPN services in our study were found to have insecure OpenVPN configurations.

A more sinister explanation for the configuration weaknesses might be that some access-network firewalls are intentionally configured to permit insecure VPN protocols and applications while blocking ones that are not vulnerable. This encourages commercial VPNs to support vulnerable but apparently more reliable settings and protocols, which can then be spied upon. There is anecdotal evidence, both personal experience and online discussions about traveling in China and other countries, of this kind of practice in regions with strict government surveillance of citizens. For example, PPTP often works while OpenVPN does not, even though PPTP is not technically any more difficult to block. We can only speculate about the exact reason for such selective blocking.

8 Related Work

There are have been various user de-anonymization attacks that leak information about the VPN user [12]. In a majority of the cases, the VPN reveals user information due to IPv6 traffic and DNS leakage [30]. Fazal et al. demonstrated that an attacker could penetrate into the VPN tunnel by exploiting clients with a dual-NIC that supports both Wi-Fi and Ethernet [16]. Similarly, privilege escalation attacks [9,10] allowed an attacker to gain access over the VPN traffic.

VPN protocols have undergone critical cryptanalysis in the past. Among others, MS-CHAPv2 in PPTP is known to have cryptographic weaknesses, and it is possible to break the protocol with exhaustive key search [22,31]. If used against a large number of users and connections, these attacks demand relatively heavy computing resources, whereas the configuration flaw explained in this paper immediately exposes the user traffic. Another known issue of PPTP is that the configuration packets in the PPP negotiation phase are not authenticated [26]. This allows a network attacker to spoof the packet containing the DNS server's address and effectively force all name resolution to happen through a compromised name server. On the other hand, the PPTP misconfiguration issue that we discuss in this paper allows the attacker to obtain all network traffic of the victim, including DNS. Also, by forcing no encryption, the victim's traffic is visible to everyone, including the attacker.

Oracle-based attacks on VPN systems can also undermine the security of the tunneled traffic. For example, the compression oracle attack on OpenVPN compression algorithms allows an attacker to send cross-domain requests when an HTTP website is tunneled through OpenVPN connections [27]. Similarly, by exploiting Bleichenbacher oracles Felsch et al. demonstrated that reusing the same key pair across IKEv1 and IKEv2 allows an attacker to bypass authentication as well as perform impersonation attacks [17].

9 Conclusion

In this work, we analyzed the security of how popular commercial VPN providers set up, or instruct their users to set up, desktop VPN clients. We studied commonly used VPN protocols and software on Windows, macOS, and Ubuntu. We found vulnerabilities in the client configurations of most of the protocols and clients. These vulnerabilities allow network attackers to perform MitM or server impersonation on the connection and thus obtain the victim's original network traffic. Similarly, local attackers can exploit vulnerabilities to steal user credentials for the VPN services. We provide guidelines for fixing these vulnerabilities.

Our main message is that security flaws, either accidental or intentional, are not always deeply hidden in the code or cryptography. Instead, configuration mistakes, poor instructions, insecure default values, and failures to disable broken legacy features can result in widespread security issues across an entire industry.

References

1. EasySSTP. https://mac.softpedia.com/get/Network-Admin/EasySSTP.shtml
2. Identity parsing in StrongSwan. https://wiki.strongswan.org/projects/strongswan/wiki/IdentityParsing
3. Known L2TP/IPsec preshared keys. https://gist.github.com/kennwhite/1f3bc4d889b02b35d8aa
4. OpenVPN. https://openvpn.net/
5. OpenVPN management interface notes. https://openvpn.net/community-resources/management-interface/
6. SoftEther VPN project. https://www.softether.org/

7. sstp-client. https://sourceforge.net/projects/sstp-client/
8. Strongswan. https://www.strongswan.org/
9. CVE-2018-3952 (2018). https://nvd.nist.gov/vuln/detail/CVE-2018-3952
10. CVE-2018-4010 (2018). https://nvd.nist.gov/vuln/detail/CVE-2018-4010
11. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible authentication protocol (EAP). RFC 3748 (2004)
12. Appelbaum, J., Ray, M., Koscher, K., Finder, I.: vpwns: virtual pwned networks. In: USENIX Workshop on Free and Open Communications on the Internet. USENIX Association (2012)
13. Atkinson, R., Kent, S.: Security architecture for the Internet protocol. RFC 4301 (1998)
14. Bui, T., Rao, S.P., Antikainen, M., Bojan, V.M., Aura, T.: Man-in-the-machine: exploiting ill-secured communication inside the computer. In: USENIX Security 2018. USENIX Association (2018)
15. Cisco: Introduction to Cisco IPsec technology. https://www.cisco.com/c/en/us/td/docs/net_mgmt/vpn_solutions_center/2-0/ip_security/provisioning/guide/IPsecPG1.html
16. Fazal, L., Ganu, S., Kappes, M., Krishnakumar, A.S., Krishnan, P.: Tackling security vulnerabilities in VPN-based wireless deployments. In: ICC (2004)
17. Felsch, D., Grothe, M., Schwenk, J., Czubak, A., Szymanek, M.: The dangers of key reuse: practical attacks on IPsec IKE. In: USENIX Security 2018. USENIX Association (2018)
18. Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., Zorn, G.: Point-to-point tunneling protocol (PPTP). RFC 2637 (1999)
19. Horst, M., Grothe, M., Jager, T., Schwenk, J.: Breaking PPTP VPNs via RADIUS encryption. In: Foresti, S., Persiano, G. (eds.) CANS 2016. LNCS, vol. 10052, pp. 159–175. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48965-0_10
20. Hurst, R., Palekar, A.: Microsoft EAP CHAP extensions. IETF Draft (2007)
21. Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., Kivinen, T.: Internet key exchange protocol version 2 (IKEv2). RFC 7296 (2014)
22. Marlinspike, M., Ray, M.: Divide and conquer: Cracking MS-CHAPv2 with a 100% success rate (2012). https://www.cloudcracker.com/blog/2012/07/29/cracking-ms
23. Microsoft: Routing and remote access service. https://docs.microsoft.com/en-us/windows/desktop/RRAS/
24. Microsoft: RRAS's VpnStrategy. https://msdn.microsoft.com/en-us/library/ee808236.aspx
25. Microsoft: Secure Socket Tunneling Protocol (SSTP). https://msdn.microsoft.com/en-us/library/cc247338.aspx
26. Mudge, Schneier, B.: Cryptanalysis of microsoft's point-to-point tunneling protocol (PPTP). In: Proceedings of the 5th ACM Conference on Communications and Computer Security. ACM Press (1998)
27. Nafeez, A.: Compression Oracle attacks on VPN networks. Blackhat, USA (2018)
28. Pall, G., Zorn, G.: Microsoft point-to-point encryption (MPPE) protocol. RFC **3078** (2001)
29. Pereira, R., Beaulieu, S.: Extended Authentication within ISAKMP/Oakley (XAUTH). IETF Draft (1999)
30. Perta, V.C., Barbera, M.V., Tyson, G., Haddadi, H., Mei, A.: A glance through the VPN looking glass: IPv6 leakage and DNS hijacking in commercial VPN clients. In: Proceedings on Privacy Enhancing Technologies (2015)

31. Schneier, B., Mudge, Wagner, D.: Cryptanalysis of Microsoft's PPTP authentication extensions (MS-CHAPv2). In: Secure Networking–CQRE. LNCS, vol. 1740, pp. 192–203. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-46701-7_17
32. Simpson, W.: The point-to-point protocol (PPP). RFC **1661** (1994)
33. Wood, D., Stoss, V., Chan-Lizardo, L., Papacostas, G.S., Stinson, M.E.: Virtual private networks. In: International Conference on Private Switching Systems and Networks (1988)

Securing Optical Networks by Modern Cryptographic Techniques

Joo Yeon Cho(✉)

ADVA Optical Networking SE, Fraunhoferstrasse 9a, 82152 Martinsried, Germany
jcho@advaoptical.com

Abstract. The deployment of optical networks grows due to the demand of high-speed communication in our daily life. Virtually all modern security solutions on optical networks are based on cryptographic technology. Various security solutions using cryptographic techniques have been adapted in current optical networks. In this paper, we investigate advanced cryptographic techniques to ensure the long-term security of optical networks.

Keywords: Optical network security · Quantum security · Authentication · Software-defined network

1 Introduction

Optical communication systems have capability to transport a large amount of data at extremely high speed (currently up to 600 Gbit/s) with very low latency. High-bandwidth applications such as data center interconnection and 5G networks deploy optical networks for transporting large data in a long distance. Optical networks are one of the core infrastructure to enable today's high speed internet and mobile communication.

Traditionally optical networks have been configured and controlled by a centralized network management system. Since modern optical networks are getting complex and handle huge amount of data, network operators have difficulties to manage their networks only by a simple management system. Recently a framework of software defined networking (SDN) is introduced to optical networks to solve such difficulties in a flexible and cost-effective way [29]. Though SDN-based network control is not yet widely deployed in existing optical networks, SDN is likely to become a dominant framework for optical network control in the near future.

While the SDN paradigm enables the deployment of centralized and integrated security policies, which simplifies complicated solutions of network security problems, a centralized optical network management has several downsides. A single-point failure is the most critical risk. A malicious behavior or a misuse of the SDN controller may interrupt the entire network services. Also, a centralized network structure is potentially vulnerable to DDOS (Distributed Denial of Service) attacks [27]. Those downsides can be compensated by integrating a decentralized authentication mechanism into the network management.

Data encryption over an optical fibre channel is another must-have feature in optical networks. A standardized encryption algorithm such as AES-256 has been widely used for protecting data during transmission. Usually an AES key is established via an authenticated key exchange protocol using standard public-key crypto algorithms such as RSA or (EC)DH. However, it is known that classical public-key cryptosystems could be broken by Shor's algorithm in a polynomial time when quantum computers are available [28]. Nowadays quantum attacks are considered as a critical threat against the long-term security of optical network infrastructure.

1.1 Our Contribution

This paper investigates how the modern cryptographic techniques can be used to reduce current risks of the optical network security. We focus on following two topics: quantum-resistant data encryption and Hash-based authentication.

Quantum-Resistant Encryption. It is known that Grover's algorithm can achieve only quadratic speedup of brute-force attack against symmetric key encryption [10]. Hence, the 128-bit quantum security can be still achieved if an AES-256 algorithm is used for data encryption. Whereas, classical ephemeral key exchange protocols such as RSA or (EC)DH should be replaced by a quantum-resistant algorithm. We propose a McEliece-based key establishment mechanism which is known to be secure against quantum attacks [20]. Even though a key size is quite large, the security level of the McEliece system has remained remarkably stable, despite dozens of attack papers over 40 years [2]. Other quantum-resistant key exchange schemes using a smaller size of a key might provide better performance. However, they could not provide as strong confidence as the McEliece cryptosystem. Note that the McEliece-based key exchange can be combined with a classical key exchange for the smooth migration to the quantum security. This is so-called a hybrid mode key exchange.

Hash-Based Authentication. The proposed scheme consists of two stages. One stage is to authenticate a SDN controller and optical node controllers (upper level). The other stage is to authenticate among a node controller and optical components (lower level).

A mutual authentication between a SDN controller and optical nodes can be achieved by a popular secure communication protocol such as TLS (Transport Layer Security). However, due to the evolving nature of open protocols and the technical difficulties, TLS is often not fully implemented nor activated in the SDN controller [24]. We propose a lightweight hash-based authentication scheme for software-define optical networks. In order to keep the efficient framework of SDN, the upper level takes a centralized authentication mechanism, which enables a fine-grained authentication for orchestrators, applications, and logging in the context of SDN.

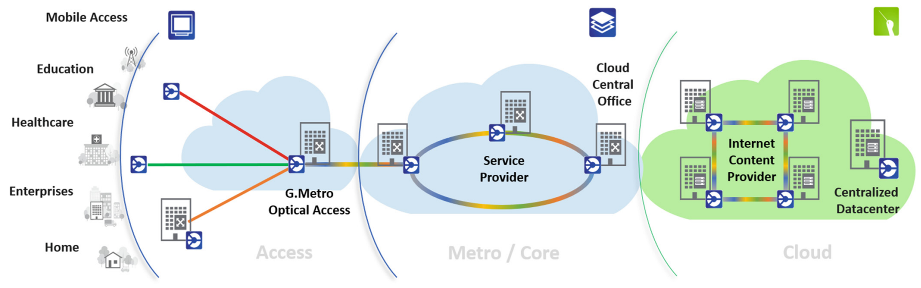


Fig. 1. A popular deployment of optical networks

On the other hand, actual optical communications between optical components are carried out in a peer-to-peer way, so that a decentralized authentication scheme is suitable for the lower level. The proposed scheme is based on the Merkle signature scheme (MSS), which is recently extended to several hash-based signature schemes (HSS) [12,21,22]. A benefit of HSS is that it does not require computationally expensive mathematical operations such as modular exponentiation or discrete logarithm. Hence, HSS seems to be suitable for authenticating optical components which are resource constrained devices. Moreover, a hash-based signature is known to be resistant to any known attacks including quantum attacks, which makes the proposed scheme suitable for long-term security.

We note that we do not propose a new quantum-resistant encryption scheme or a hash-based authentication mechanism. Our contribution is to integrate such mechanisms on a SDN-based optical network, analyze their benefits and disadvantages, and compare them with the standard mechanisms based on traditional cryptographic techniques.

The rest of this paper is structured as follows: first, we describe a brief structure of the optical network system and identify major security threats against optical networks. Next, we introduce currently available security options. We investigate the problems of such options and propose new solutions based on modern cryptographic techniques. Finally, we conclude the paper.

2 System Description

Optical communications are performed over the optical fiber layer (or Layer 1). Using wavelength-division multiplexing (WDM) technology, a number of optical carrier signals can be multiplexed onto a single optical fiber by using different wavelengths of laser light. The transmitted wavelengths are demodulated and detected in a receiver side. The specific optical transmission and switching characteristics, such as circuit, burst, and packet switching on wavelength channels, pose challenges for controlling optical networks. For operator's usability, an optical network management such as system configuration, monitoring and maintenance can be done over an IP network. Note that such IP network is usually isolated from public networks to avoid malicious attacks from external network. Figure 1 shows a popular deployment of optical networks. Optical networks have several unique features. A node of the optical network is an integration of a node controller and multiple optical components such as transceivers, amplifiers and switches. A node controller is used to configure and control components, while optical communications are carried out between transceivers in peer-to-peer way. Optical components are usually hot pluggable, meaning that an optical component can be added and removed on the backplane while the system is being operated. The system automatically detects and recognizes the change. An interface between an optical network controller and actual optical transceivers are performed through a dedicated backplane [16]. A typical configuration of an optical network node is depicted in Fig. 2.

3 Security Threats

Security threats in optical networks are identified mainly by the following categories.

Eavesdropping. Eavesdropping is to attempt an unauthorized access to the carried data for the purpose of stealing data or analyzing the network traffic without breaking the connection. It is not very difficult to intercept an optical signal: One can tap the fiber using a commercially available clip-on coupler that can detect the leaked optical signal caused by a bend in the fiber. A more complex method is to observe the signal leaked due to crosstalk in optical switching and perform eavesdropping [30]. Fiber cut or re-routing channel would be similar unauthorized attempts to access data over fiber. Several fiber-optic tapping incidents have been already reported in the press.

Optical Disrupting. This attack is to degrade or completely disrupting service on an optical layer via optical link cuts, signal insertion, or signal splitting. For instance, high-power jamming, amplifier-transient or mixed-modulation attacks can cause a series of harmful effects to the co-propagating user signals inside optical fibers, amplifiers, and switches, resulting in reduced optical signal-to-noise ratio (OSNR) and degraded bit error ratio (BER) [8].

(Distributed) Denial of Service Attack. Denial of Service (DoS) attack is one of the critical cyber-attacks on SDN-based optical networks. Attackers can launch DoS attacks on the user plane by sending bogus packets to the optical network. The path towards the core network can be flooded by bogus packets. This would lead to a denial of service or at least a throughput degradation caused by the congestion to networks. A simulation result of DDoS attacks on the optical fiber cable is presented in [18].

Network Intrusion. This attack aims to intrude a network, access resources, and manipulate the network operation. Malicious applications and network devices may allow an attacker to introduce vulnerabilities to the system. An SDN-based network management is particularly vulnerable to this type of attack because the attacker can control the entire network system by hijacking a SDN controller. Hence, both an authentication and an access control mechanism need to be implemented properly to avoid such unauthorized access to the network.

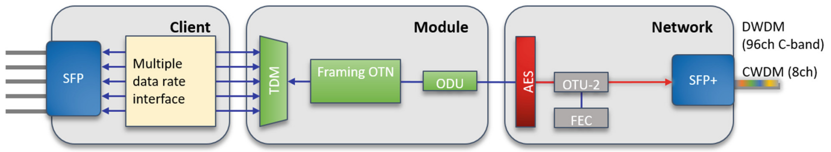


Fig. 2. A typical configuration of an optical network node

Quantum Attacks. Quantum attacks are a new but a critical threat against the internet security. It is known that most of popular public-key cryptosystems, such as RSA, ECC and Diffie-Hellman will be broken by Shor's algorithm [28] when a large scale of quantum computers would be available. One may argue that it may be too early to discuss quantum threat at this stage since no one knows when quantum computers can be actually built. However, as long as there exists a non-negligible risk of a harvest-type attack (a store-now-decrypt-later attack) and optical networks are usually built for the critical infrastructure, quantum security became one of the important topics that optical networks should consider for a long-term security Fig. 2.

4 Current Security Solutions

Building secure optical networks is challenging due to unique features of optical networks. User data are framed into G.709 Optical Transport Network (OTN) format and transported over Layer 1 [9]. Popular security protocols such as TLS/SSL or IPsec do not apply here. Data encryption should be done in hardware such as ASIC or FPGA due to the requirement of high speed and low latency data transport. Note that a hardware-based data encryption needs a special care to catch up the high speed data rate. Optical communication is always bidirectional, meaning that encryption and decryption are performed in parallel at each node, which requires double crypto engines.

Similarly to other classes of networks, optical networks focus on three major security topics: data confidentiality, user/device authentication and data integrity.

Data Confidentiality. The primary concern on the confidentiality is on being eavesdropped by fiber tapping. Due to high bandwidth of an optical channel, tapping fiber in a very short time can lead a large amount of data loss. Hence, user data are commonly encrypted by a symmetric encryption algorithm such as AES (Advance Encryption Standard) algorithm. User data are encrypted and inserted into the payload field of the G.709 OTN framework, as shown in Fig. 3. A data encryption key (an AES key) should have a lifetime. A re-key interval should be carefully considered on the high-speed link since the security level is bounded by the maximum amount of date that is encrypted with a single key. A specific value of the key lifetime should be determined in accordance with some safety margin for protocol security. In [19], it is given that low bounds on the

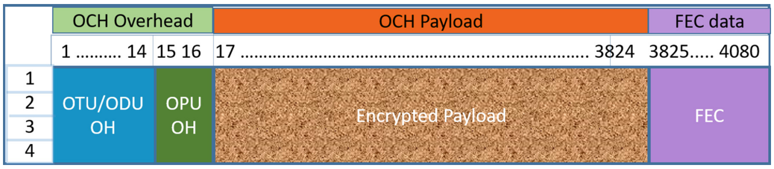


Fig. 3. G.709 Optical Transport Network Frame

amount of data that AES-GCM can process without needing a key change. These bounds are also rephrased in the specification of TLS 1.3 [26]. Unfortunately, there is no dedicated field in G.709 frame that can be used for a key exchange protocol. An industrial de-facto solution is to abuse some of (unused) overhead field to perform a key exchange protocol.

Authentication. In terms of network security, authentication is divided into two sub-categories: user authentication and device authentication. Similarly to normal operating systems, user authentication is usually done by verifying passwords, that is, by checking a user knowledge, while device authentication can be done by checking a piece of data that device is supposed to hold in memory or a smart card.

In a nutshell, when a new optical component is added to the system, this component is verified by the host server through an authentication procedure. Popular internet security protocols such as TLS/SSL or IPSec do not apply for optical networks in general due to the constraint of computing resources. Optical components can be authenticated by a stored password or their certificates from PKI (public-key infrastructure) structure.

Since any secret that is installed in devices would be a target for various attacks, it is preferred that a device has only public information that include a signature signed by a central authority. In this context, the centralized and integrated authentication process simplifies the solution of complex network security problems. On the other hand, a malicious behavior or misuse of a centralized server may cause the disruption of entire network. This concern leads to the concept of decentralized authentication. In the next section, we will propose a two-stage authentication mechanism suitable for optical networks.

Integrity. According to G.709 framework, erroneous or modified bits in the payload can be corrected by the FEC (Forward Error Correction) up to the capability of code. If an error occurs beyond the correcting capacity, it will be detected by checksum and the packet will be dropped. In addition, a message authentication code (MAC) is generated from a payload using a secret key to ensure that the payload has not been modified during transmission. Galois/Counter Mode (GCM) is a widely adopted MAC mechanism in combination with the AES encryption algorithm (AES-GCM).

5 Quantum-Resistant Encryption

The cryptographic tools are at the center of protecting our information assets from security threats. Unfortunately, quantum computers become a serious threat on optical network security, as they can break most of current public-key cryptographic algorithms. To keep ahead of the hackers, research on quantum-safe security is actively performed around the world, as this solution renders information secure in an era with quantum computers.

Post-quantum Cryptography. Post-quantum cryptography is the most important research topic on quantum-safe security, as it aims to develop efficient quantum-resistant crypto algorithms that could replace classical crypto algorithms in quantum world. The National Institute of Standards and Technology (NIST) in the United States initiated a standardization process of post-quantum cryptography in 2016 and expects a draft standard available around 2022/2024 [5].

In general, post-quantum cryptographic algorithms fall into the following categories: lattice-based, code-based, hash-based, multivariate and SIDH (Supersingular Isogeny Diffie–Hellman). Among those, we focus on code-based crypto algorithms since their security has been studied for a long time in many aspects and, therefore, it is unlikely hidden attacks are found in the near future. In addition, the theory of error correction codes itself is well developed and understood.

The general idea behind code-based crypto schemes is to first select a code for which an efficient decoding algorithm is known, and then to create a trapdoor function by disguising the code as a general linear code. Since the problem of decoding a linear code is NP-complete, a description of the original code serves as the secret key, while a description of the transformed code serves as the public key.

McEliece Cryptosystem. The first code-based crypto scheme was introduced by McEliece in 1978. Since then, the McEliece cryptosystem has been extensively analyzed for more than thirty years, and its original form using Goppa codes is still unbroken. It became now the basic construction of code-based cryptosystems. Several other underlying codes have been proposed so far, but most of them are broken [25]. Due to long-lasting security and well-developed theory, the McEliece cryptosystem using a binary Goppa code would be a good cryptographic primitive that can be used for a key exchange scheme.

Due to this background, we propose a post-quantum key exchange scheme based on McEliece public key cryptosystem using a binary Goppa code. It is known that the McEliece cryptosystem is very efficient at encryption and decryption, but suffers from very large key sizes. However, the McEliece cryptosystem is applicable to the use case of high-speed optical communication since the benefits of security and speed are more important than the costs of communicating and storing keys.

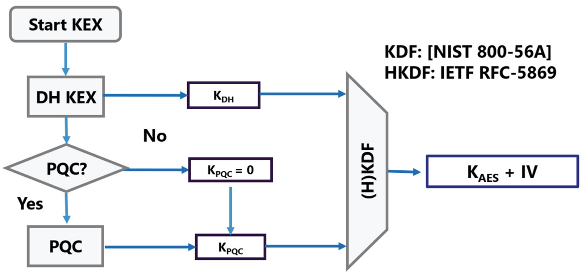


Fig. 4. A example of the key combination for hybrid mode: DH and PQC

Re-key Period. It is important to refresh a data encryption key (a symmetric encryption key) regularly within a short time period. The reason is that common cryptographic attacks are based on the ability to collect encrypted data under a single key and the amount of transmitted data over high-speed optical networks reaches quickly the maximum threshold number of messages that can be safely encrypted.

Hybrid Mode Key Exchange Protocol. A hybrid mode is a migration technique to move to quantum-resistant encryption in advance of establishing complete security assurance in quantum computer world. There are several methods for deriving cryptographic keys from two shared secrets; the two raw shared secrets (the ECDH shared secret and the PQC shared secret) can be concatenated or XORed to derive a secret key, or putting them through a KDF (key derivation function) to derive a common shared secret, as shown in Fig. 4. The shared secrets are established using existing classical key agreement schemes such as ECDH, and quantum-resistant key encapsulation methods (PQ-KEMs). The key derivation functions for the hybrid mode key agreement can be taken from e.g. NIST SP 800-56C [23] or IETF RFC-5869 [17].

Implementation. We implemented a McEliece-based key exchange scheme with a very conservative parameter set, e.g. a 300-bit classical security corresponding to a 150-bit post-quantum security, on optical networks. Even though the size of a public key is quite large, i.e. around 1.3 Mbytes, the key exchange can be completed before exceeding the maximum limit of data encryption. We demonstrated a high-speed quantum-safe optical communication over the GÉANT network. The central building block of our demo system is a hybrid mode key exchange scheme combining a post-quantum key exchange (McEliece-based), together with a classical key exchange (Diffie-Hellman). We performed a hybrid mode quantum-safe key exchange over a 100G optical communication link between long-distant nodes, one was located in Poznań (Poland) and the other was in Trondheim (Norway), and connected over the NRENs network. The distance was around 2,800 km [6].

Table 1. Average execution time of NIST KEM candidates on PowerPC test platform. opt = optimized code, ref = reference code

Candidate	**keypair gen.**	**enc.**	**dec.**	**parameter**	**ref/opt**
Classic McEliece	43 s	23 ms	28 ms	mceliece8192128	opt
Classic McEliece	226 s	53 ms	3129 ms	mceliece8192128	ref
NTS-KEM	6.6 s	21 ms	54 ms	NTS-KEM(13, 136)	opt
HQC	0.045 s	98 ms	127 ms	Paranoiac-IV	opt
HQC	0.053 s	114 ms	167 ms	Paranoiac-IV	ref
LEDAcrypt	0.602 s	78 ms	882 ms	SL=5 N0=2	opt
LEDAcrypt	1.1 s	217 ms	2438 ms	SL=5 N0=2	ref

We extended our experiments for other code-based crypto algorithms. In Jan. 2019, NIST announced the 17 second-round candidate of public-key encryption and key exchange algorithms. Among those, there are 7 code-based KEMs, including Bike, Classic McEliece, HQC, LEDAcrypt, NTS-KEM, ROLLO and RQC. We implemented these code-based KEMs on our platform and evaluated the feasibility and their performance. Details are given in Table 1. Note that Bike, ROLLO and RQC have not been evaluated because their codes required external libraries such as NTL and GMP which were not available on our test platform.

6 Hash-Based Authentication

Our proposal uses a hash-based signature scheme (HSS) as a basic building block of the authentication. HSS does not rely on the conjectured hardness of mathematical problems. Instead, its security assumptions rely on those of a cryptographic hash function: pre-image resistance and collision resistance. No random oracle or number-theoretic assumptions are required.

6.1 Hash-Based Signature

A Hash-based signature was initially proposed by Merkle in the late 1970s [22]. Since then, several variants of the Merkle signature have been proposed [3,12,21]. Hash-based signature schemes generally feature small private and public keys as well as fast signature generation and verification but large signatures and relatively slow key generation. Unlike most other signature systems, it is known that hash-based signatures can withstand attacks using quantum computers [5].

Merkle signature scheme (MSS) consists of a one-time signature scheme (OTS) and a Merkle hash tree [22]. An OTS scheme can sign only one message with one key. A Merkle hash tree enables us to sign multiple messages with a single public key. An example of a Merkle signature scheme is shown in Fig. 5. The leaves are the hash values of public keys of OTS. Each inner node

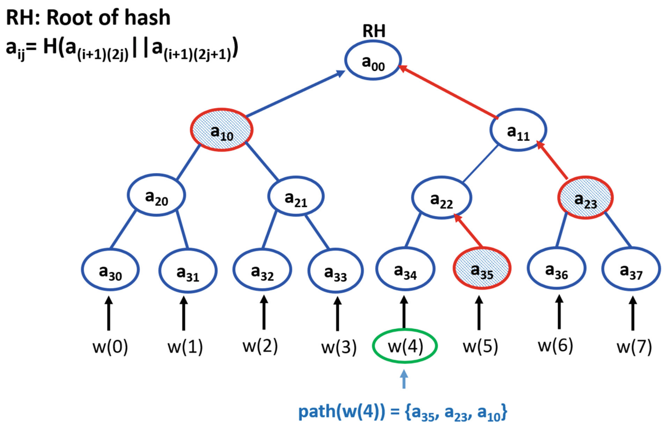


Fig. 5. Merkle-tree with a height $h = 3$. Red-colored nodes and arrows indicate the path from leaf a_{34} to the root a_{00}. (Color figure online)

is computed as the hash of the concatenation of its two children nodes, i.e., $a_{ij} = H(a_{(i+1)(2j)}||a_{(i+1)(2j+1)})$ where H is a hash function and $||$ is a concatenation.

To sign a message $w(i)$, a signer first computes a constant-sized digest $d(i) = H(w(i))$. Then, the signer generates a one-time signature $\sigma(w(i))$ of the digest using the i-th OTS key $sk(i)$. Note that i is the index chosen from $\{0, \ldots, 2^h - 1\}$ and h is the height of the tree. Since each OTS key should be used only once, the management of the key index is an important issue for the security of OTS schemes.

Let $S(\cdot)$ denote a Merkle signature of the $w(\cdot)$. Then, $S(w(i)) = \{i, \sigma(w(i)), pk(i), path(i)\}$, where the index i, the OTS signature $\sigma(w(i))$, the i-th OTS public key $pk(i)$, and the $path(i)$ to the root of tree. The $path(i)$ consists of the siblings for all nodes along the path from the i-th leaf to the root. In the example shown in Fig. 5, $path(4) = \{a_{35}, a_{23}, a_{10}\}$.

To verify the signature S on message $w(i)$, the verifier first validates the OTS signature on the message by $pk(i)$. Then, a root value of the Merkle tree is computed using the nodes in $path(i)$. If this root value matches the one given as the public key, the signature is accepted, otherwise it is rejected.

Advanced Hash-Based Signature Schemes. Since Merkle invented a hash-based signature in 1979, several variants have been proposed to improve the efficiency and performance: XMSS (eXtended Merkle Signature Scheme) [12], XMSSMT [14], HSS (Hash-Based Signatures) [21], SPHINCS [3], and SPHINCS+ [1]. In particular, XMSS and HSS have been proposed in the Internet Engineering Task Force (IETF). We omit the detailed description of these schemes since it is straightforward to replace MSS by these schemes.

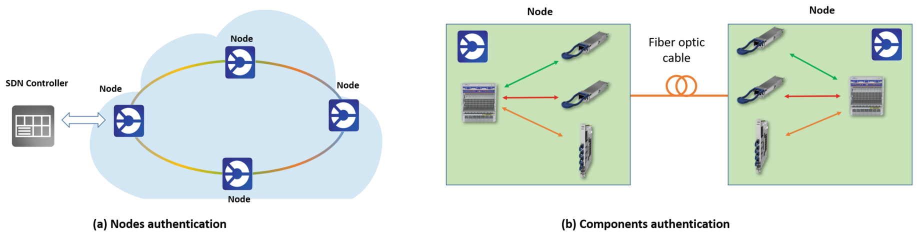


Fig. 6. A two-stage authentication in the optical network

6.2 Authentication Schemes

As mentioned in the introduction, the proposed authentication scheme consists of two stages. The first stage is an authentication between an SDN controller and optical nodes for the trusted management (Fig. 6(a)), while the second stage is between components for the trusted optical communication (Fig. 6(b)). This two-staged authentication is necessary because a network management and actual optical communications are separated in the optical network. Optical components perform a mutual authentication directly during transmission in order to minimize the latency.

Node Authentication. Suppose an optical network is composed of n nodes that are managed by a SDN controller. The SDN controller chooses the height of Merkle tree h where $0 < n \leq 2^h - 1$ and other security parameters. The height h determines the size of the Merkle signature.

The SDN controller creates a certificate $Cert_{node}$ for each node. The certificate consists of a data field and its signature where a data field contains a random number, a Merkle signature, and a signature given by the SDN controller. This certificate is used for the node authentication.

$$Cert_{node} = \{r, S(r), Sign_{SDN}\}$$

The certificate management such as enrollment, renewal, and revocation can be done by an internet protocol such as SCEP (Simple Certificate Enrollment Protocol) [11].

Component Authentication. Each node controller creates its own Merkle signature where the size of Merkle tree can be different, depending on the number of components of each node. Since each node is usually located remotely on an optical network, it is important to verify a public key of Merkle signature that a counterpart node uses. One may use a PKI infrastructure. However, a centralized key validation approach has several downsides, as mentioned in the introduction.

Our proposal is to use a blockchain structure. A root value (a public key) of Merkle tree is hash-chained, together with meta data of the node and a hash value of the previous block, as shows in Fig. 7. This hash chain is distributed

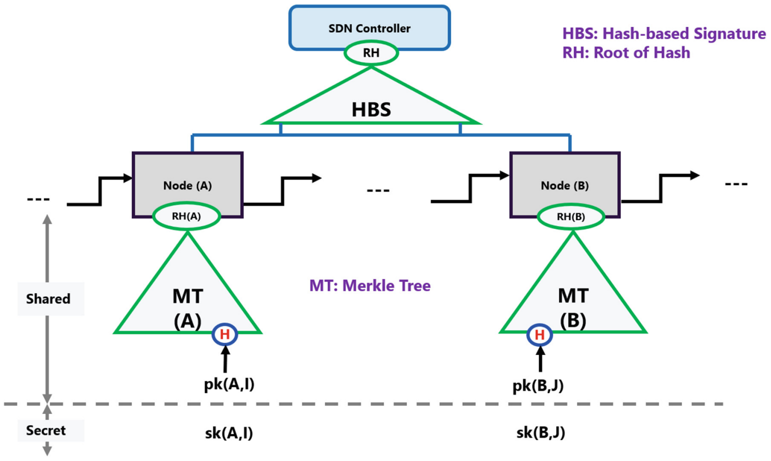


Fig. 7. Structure of authentication using hash-based signature over software-based optical network

by the SDN controller and installed on each node controller. It is infeasible to modify the hash chain unless the SDN controller is corrupted. The update of the hash chain can be done only by the SDN controller via a secure protocol. The last block of hash chain is a trusted block which is added by the SDN controller, otherwise the last block of hash chain would not be validated.

By this structure, any public key of Merkle signature can be verified without a centralized authentication server. Note that, even though our proposal is similar to blockchain, a signing process itself is not chained, hence, it does not validate the occurrence of signing event itself. Rather, it provides the trust of public key in a decentralized manner so that it reduces the risk of a single point failure.

Practical Aspects. The practicability of hash-based signature schemes has been demonstrated in several papers, e.g., see [3,4,7,12,15]. There, the hash-based signatures have been implemented on constrained devices such as an 8-bit AVR and a 16-bit smart card at a speed comparable to RSA and ECDSA.

Suppose we choose a XMSS scheme with a tree height $H = 20$ and $n = 32$ (e.g. SHA-2-256). This parameter set allows to generate $2^{20} \approx$ one million signatures with the security level of 2^{236} [13]. Then, the sizes of a public-key, a secret key, and a signature are around 2.2 kB, 1.3 kB, and 2.8 kB, respectively. According to [13], a signing process, fully implemented in C language, takes 3.24 ms on a 3.5 GHz Intel i7 CPU platform.

7 Conclusion and Future Plan

Optical networks are evolving. Various cryptographic techniques have been used for the security of optical networks. We showed that modern cryptographic techniques such as quantum-resistant encryption or hash-based signature are able to improve further the security of optical networks. We have already implemented and performed a field-test for quantum-resistant encryption over optical network. Since the standardization process of post-quantum cryptography is still going on, our evaluation would subject to changing by upcoming report from NIST. However, based on our experience that there is a great timing gap between the standardization and the real deployment, our evaluation would contribute to minimize such gap. As a next step, we will implement the proposed mechanisms on a SDN controlled optical network and check the performance and feasibility. Since several detailed aspects are missing, this would be filled by experimental implementations.

References

1. Bernstein, D., et al.: SPHINCS+: submission to the NIST post-quantum project (2017)
2. Bernstein, D., et al.: Classic McEliece: conservative code-based cryptography (2019). https://classic.mceliece.org/nist/mceliece-20190331.pdf
3. Bernstein, D.J., et al.: SPHINCS: practical stateless hash-based signatures. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 368–397. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_15
4. Hülsing, A., Busold, C., Buchmann, J.: Forward secure signatures on smart cards. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 66–80. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35999-6_5
5. Chen, L., et al.: Report on post-quantum cryptography, NISTIR 8105 (2016)
6. Cho, J., et al.: Demonstration: field test of high speed quantum-safe optical communication over GÉANT. In: TNC 2018, Trondheim, June 2018. https://tnc18.geant.org/core/event/96
7. Eisenbarth, T., von Maurich, I., Ye, X.: Faster hash-based signatures with bounded leakage. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 223–243. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_12
8. Furdek, M., Skorin-Kapov, N.: Physical-layer attacks in all-optical WDM networks. In: 2011 Proceedings of the 34th International Convention MIPRO, pp. 446–451, May 2011
9. Gorshe, S.: A tutorial on ITU-T G.709 optical transport networks (OTN)
10. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC 1996, pp. 212–219. ACM (1996)
11. Gutmann, P.: Simple certificate enrollment protocol, draft-gutmann-scep-10, March 2018
12. Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., Mohaisen, A.: XMSS: extended hash-based signatures, Internet-Draft draft-irtf-cfrg-xmss-hash-based-signatures-12, Internet Engineering Task Force, Work in Progress, January 2018
13. Hülsing, A.: Hash-based signatures. Summer School on Post-Quantum Cryptography, Eindhoven, Netherlands, June 2017
14. Hülsing, A., Rausch, L., Buchmann, J.: Optimal parameters for $XMSS^{MT}$, Cryptology ePrint Archive, Report 2017/966 (2017). https://eprint.iacr.org/2017/966

15. Hülsing, A., Rijneveld, J., Schwabe, P.: Armed SPHINCS – computing a 41 kB signature in 16 kB of RAM, Cryptology ePrint Archive, Report 2015/1042 (2015). https://eprint.iacr.org/2015/1042
16. IEEE, IEEE 802.3 ethernet working group. http://www.ieee802.org/3/
17. Krawczyk, H., Eronen, P.: HMAC-based extract-and-expand key derivation function (HKDF) (2010). https://tools.ietf.org/html/rfc5869
18. Kumar, S.: Simulating DDOS attacks on the us fiber-optics internet infrastructure. In: Proceedings of the 2017 Winter Simulation Conference (2017). https://www.informs-sim.org/wsc17papers/includes/files/477.pdf
19. Luykx, A., Paterson, K.: Limits on authenticated encryption use in TLS. www.isg.rhul.ac.uk/~kp/TLS-AEbounds.pdf
20. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Deep Space Netw. Progr. Rep. **44**, 114–116 (1978)
21. McGrew, D., Curcio, M., Fluhrern, S.: Hash-based signatures (2018). https://tools.ietf.org/html/draft-mcgrew-hash-sigs-10
22. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21
23. NIST, Recommendation for key-derivation methods in key-establishment schemes, Special Publication 800-56C, Rev. 1 (2018)
24. OpenFlow, Openflow switch specification, Version 1.5.0, ONF TS-020 (2014)
25. Overbeck, R., Sendrier, N.: Code-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 95–145. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7_4
26. Rescorla, E.: The transport layer security (TLS) protocol version 1.3, Internet-Draft draft-ietf-tls-tls13-12, March 2016
27. Natarajan, S., Scott-Hayward, S., Sezer, S.: A survey of security in software defined networks. IEEE Commun. Surv. Tutor. **18**(1), 623–654 (2016)
28. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, SFCS 1994, pp. 124–134. IEEE Computer Society (1994)
29. Thyagaturu, A.S., Mercian, A., McGarry, M.P., Reisslein, M., Kellerer, W.: Software defined optical networks (SDONs): a comprehensive survey. IEEE Commun. Surv. Tutor. **18**(4), 2738–2786 (2016)
30. Zafar Iqbal, M., Fathallah, H., Belhadj, N.: Optical fiber tapping: methods and precautions. In: 8th International Conference on High-capacity Optical Networks and Emerging Technologies, pp. 164–168, December 2011

Platform Security and Malware

A Roadmap for Improving the Impact of Anti-ransomware Research

Jamie Pont, Osama Abu Oun, Calvin Brierley, Budi Arief(✉), and Julio Hernandez-Castro

University of Kent, Canterbury, UK
{jjp31,oa354,crb34,ba284,jch27}@kent.ac.uk

Abstract. Ransomware is a type of malware which restricts access to a victim's computing resources and demands a ransom in order to restore access. This is a continually growing and costly threat across the globe, therefore efforts have been made both in academia and industry to develop techniques that can help to detect and recover from ransomware attacks. This paper aims to provide an overview of the current landscape of Windows-based anti-ransomware tools and techniques, using a clear, simple and consistent terminology in terms of *Data Sources*, *Processing* and *Actions*. We extensively analysed relevant literature so that, to the best of our knowledge, we had at the time covered all approaches taken to detect and recover from ransomware attacks. We grouped these techniques according to their main features as a way to understand the landscape. We then selected 15 existing anti-ransomware tools both to examine how they fit into this landscape and to compare them by aggregating their accuracy and overhead – two of the most important selection criteria of these tools – as reported by the tools' respective authors. We were able to determine popular solutions and unexplored gaps that could lead to promising areas of anti-ransomware development. From there, we propose two novel detection techniques, namely serial byte correlation and edit distance. This paper serves as a much needed roadmap of knowledge and ideas to systematise the current landscape of anti-ransomware tools.

Keywords: Ransomware · Anti-ransomware · Detection · Recovery

1 Introduction

Ransomware, a type of malware used to extort money from victims, has existed in various forms since the 1980s [1] and has incorporated more sophisticated features since 1996 when the idea of cryptoviral extortion was first introduced [2]. Throughout the years, there have been various types of ransomware including *device lockers* and *crypto-ransomware* [3]. Device lockers restrict access to a device by locking the screen (without encrypting any data) and displaying a ransom note. On the other hand, crypto-ransomware encrypts the victim's files such that a corresponding decryption key is required to regain access. In all cases, the victim is typically notified through the use of a ransom note often accompanied by threatening demands and instructions on how to pay (usually via cryptocurrency such as Bitcoin). The attacker will only release the decryption key if the ransom is paid.

Unfortunately, individuals and organisations are still frequently hit by ransomware attacks that cause severe disruption and substantial costs. There has also been an increase recently in targeted attacks, i.e. large-scale ransomware infections aimed at specific organisations, which effectively can bring businesses to a halt [4]. As with other types of malware attacks, it has been repeatedly shown that running up-to-date antivirus software is generally not enough to prevent ransomware attacks. Offline backups are the only reliable security countermeasure to mitigate a ransomware attack, but unfortunately they are still not common, particularly in small and medium organisations.

Additionally, many cybercriminals simply make use of the code or ideas from other relatively successful ransomware variants in order to make a quick profit [5,6]. Also, the availability of *Ransomware-as-a-Service (RaaS)* [7] means cybercriminals can go to the underground market to purchase ransomware kits, such as Satan [8], allowing them to deploy their own ransomware variants without needing in-depth technical knowledge.

Due to the significant damage and disruption that ransomware can cause [9], there is an increasing demand for research in anti-ransomware tools and techniques. For instance, the "No More Ransom" project maintains a collection of defeated ransomware variants along with tools to help victims recover any lost data [10]. Users are also often advised to follow best practices with regard to backing up their data and dealing with unexpected links and email attachments to help mitigate the risk of a ransomware infection [11].

However, this is not enough, so a number of techniques are in development and being implemented to detect the presence of a ransomware infection quickly, with the aim of stopping it before it causes any significant damage or data loss. Similar approaches include attempting to recover any data the ransomware did manage to encrypt, to ensure that the victim experiences minimal or no disruption. We expand further on the techniques and their results in Sect. 5.

Contribution. First, we present a novel feature-based roadmap of the techniques that are commonly used in anti-ransomware tools. This is constructed based on the analysis of the state-of-the-art in anti-ransomware tools – open-source, where possible – from academic research. Second, we propose two new techniques to detect ransomware through serial byte correlation and edit distance. These are detailed in Sect. 4. We envision that our paper can help in guiding future work in anti-ransomware research by providing researchers with a single point of reference, allowing them to reason about new and existing anti-ransomware techniques.

2 Related Work

There are two types of taxonomies covering the ransomware domain: Ransomware and Anti-Ransomware. The former is quite common in the literature, whereas to the best of our knowledge, only one occurrence of the latter exists. Al-rimy et al. present a ransomware taxonomy based on three factors: *Severity*, *Platform* and *Target* [12], each of which is further sub-categorised. Ahmadian et al. present a high-level taxonomy of ransomware splitting it into two main types: *Non-Cryptographic Ransomware (NCR)* and *Cryptographic Ransomware (CGR)* [13]. CGR is further split into *Private-Key Cryptosystem Ransomware (PrCR)*, *Public-Key Cryptosystem Ransomware (PuCR)* and *Hybrid Cryptosystem Ransomware (HCR).*

In [14], Kharraz et al. analysed 1,359 ransomware samples across 15 distinct ransomware families to determine ransomware characteristics in order to help propose detection strategies. Useful insights are given, including how ransomware accesses a victim's files, how it changes the Master File Table (MFT) and how ransom payment is implemented. However, not all aspects are considered such as infection vectors nor evasive techniques. They also propose the idea of monitoring the filesystem for detecting ransomware, a technique used by many anti-ransomware algorithms today, as shown in Table 1 later.

Scaife et al. discussed two additional characteristics: *filesystem traversal preferences* and *file format attack frequency* [15]. Three types of traversal were shown: depth-first with encryption starting at the leaves, depth-first with encryption starting at the root, and extension-based. The most targeted file types were .pdf, .odt, .docx and .pptx, indicating that cybercriminals prioritise productivity-related files rather than personal files (such as pictures and videos).

Gazet presented an analysis of 15 ransomware samples across four families, providing insights into the structure of the ransomware code and the encryption schemes used [16]. The study additionally examined the extortion schemes implemented and their infection vectors, however concluded that the ransomware that was analysed was not suitable for mass extortion.

To the best of our knowledge, Al-rimy et al. [12] is the first and only published paper so far that presents an anti-ransomware taxonomy. They categorise existing research into two groups: *Analysis research* and *Counteractions research. Analysis research* investigates the behaviour of the ransomware and tries to categorise it into families. It is usually conducted in a monitored environment – mostly isolated in a research laboratory – either using *static methods* (a passive approach in which the ransomware payload would be studied without running it) or *dynamic methods* (where ransomware will be analysed during execution). The focus of *Counteractions research* is on confronting the ransomware attacks in a working environment. The authors outline three subcategories: *Prevention*, *Detection* and *Prediction. Prevention* relates to the procedures and policies aiming to protect potential victims against ransomware attacks by preventing the damage from being inflicted in the first place. *Prevention* is subdivided into *Proactive Prevention* and *Reactive Prevention. Proactive Prevention* aims to prevent the attack before it starts, while *Reactive Prevention* focuses on mitigating the effect of the attack by restoring the encrypted data. The authors define *Detection* as the process of distinguishing between malicious and benign samples. *Prediction* is presented as an early detection which enables taking preventive actions on time. These suggest that there are some inconsistencies which we feel necessary to address. Through our initial study of this taxonomy, we noticed the existence of an overlap in the definitions of *Prevention*, *Detection* and *Prediction*. There are works in the literature that might easily be classified under any of these three definitions. A more robust anti-ransomware classification system is therefore needed.

3 Methodology

Learning from Al-rimy et al. [12], our motivation was to design a landscape that avoids overlapping between categories and includes the individual anti-ransomware techniques rather than just their type. We believe that this provides a clearer and more complete overview of the methods used to defeat ransomware at a glance. We also hope that this would help other researchers catch up with the current state-of-the-art and encourage them to develop their own tools and techniques. A robust and extendable anti-ransomware classification system should:

– Clearly define current anti-ransomware techniques
– List their data sources and/or system requirements
– Compare them where possible in terms of accuracy and overhead
– Map the current state-of-the-art onto the landscape

With these criteria in mind, we first defined the scope of our analysis. Research into anti-ransomware tools and techniques has covered various platforms so far including Windows, Linux and Android [17,18], but our survey revealed that most of this work has targeted PC-based (specifically Windows) ransomware. This is justifiable, as ransomware mainly targets the Windows platform [19,20]. We therefore set PC-based techniques as the main scope for our current analysis, but we firmly believe this work could easily be expanded with techniques in use on other platforms, such as Heldroid [21] for the Android platform, in the future.

We analysed the literature looking for the implementation details of various anti-ransomware tools. Although these tools have largely similar goals (i.e. detection, recovery, prevention or a combination of those), their implementations vastly differ. Our analysis highlighted that there are two major types of anti-ransomware tools: those developed by the academic community and those developed by antivirus vendors. Whilst it was our intention to ensure that this work encompassed the anti-ransomware landscape as accurately as possible, various reasons led us to restrict the current analysis to techniques used in academic and open source software. These reasons are discussed further in Sect. 5.3.

After finding a number of similarities between the various approaches and techniques studied, we were able to identify areas of crossover that could be used for grouping at a higher level. Initially, we split the landscape according to *functionality*, i.e. what the anti-ransomware tools intend to achieve. These can be largely grouped into *detection* and *recovery* strategies.

Within this high-level classification, we then looked at the individual techniques used for detection and recovery. In order to achieve detection, some *Data Source* is required along with the *Processing* of this data. The data source used for a given detection technique may require access to *Kernel Space* (such as in Data Aware Defense [22]), *User Space* (such as in RAPPER [23]) or both (such as in UNVEIL [20]). Additionally, any results from the raw data sources or the data processing steps could optionally be fed into *Machine Learning* algorithms in order to detect subtle patterns in the data to build models to distinguish between benign and malicious behaviour (as in ShieldFS [19]).

We take a similar approach to classify the strategies in ransomware recovery. To recover from a ransomware attack, some *Data Source* is required, such as a backup or access to API calls. Depending on the chosen data source, a *Processing* step may be required before the tool is able to start the recovery process.

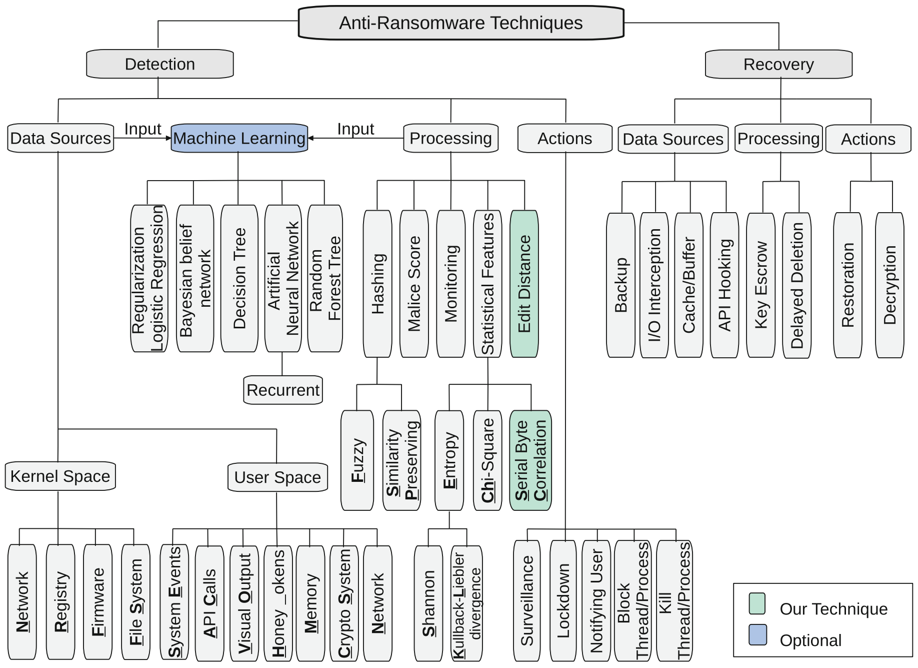


Fig. 1. An overview of the current academic anti-ransomware landscape

Our analysis of the literature also highlighted that there were several actions to react to the detection of a ransomware attack. It is common to attempt to kill or block the malicious process or thread, such as in Data Aware Defense. This often requires user confirmation to minimise false positives, such as in Redemption [24]. Recovery tools should help the user to get to a state where the effects of the attack have been alleviated, i.e. they recover access to most of the lost data. However, this does not always imply that all damage has been mitigated, due to factors such as the cost to an organisation in lost business during the downtime resulting from an attack [25].

4 Contribution

We created a roadmap of the current anti-ransomware landscape (Fig. 1), including examples of where anti-ransomware tools fit into this landscape (summarised as Table 1 in Sect. 5). We also propose novel ransomware detection techniques (serial byte correlation and edit distance) that have shown great potential in our initial experiments.

4.1 Detection

Unlike other types of malware that may wish to remain hidden for a long time, most ransomware strains usually perform encryption just after the initial infection. Once the encryption is done, it will then typically make itself known to the victim, often via a ransom note [26]. Researchers have shown that this unsophisticated behaviour can be exploited to detect the ransomware infection in its early stages. For example, as shown by UNVEIL [20], crypto-ransomware almost invariably results in obvious and repetitive I/O traces within the filesystem due to bulk encryption (which results in write and/or delete operations). Similarly, CryptoDrop [15] shows that by taking a 'data-centric' approach, i.e. focusing on modifications to user data, ransomware can also be successfully detected.

The current state-of-the-art in anti-ransomware detection aims to analyse a data source on the potential victim's system and process it in some way to decide whether or not they are under a ransomware attack. By using machine learning or some other statistical technique over this data, a decision can be made and an appropriate action taken.

Data Sources. There are several ways to collect the data required for ransomware detection. Depending on the desired approach, the data source may require access to kernel space, user space or both. In the former case, it is common to implement a *Windows Filesystem Minifilter Driver* [27]. This can provide an unrestricted view of filesystem access requests - represented as *I/O Request Packets (IRPs)*. By registering a filesystem minifilter driver with the Windows Filter Manager, it is possible to filter specific I/O requests such as reads or writes. The IRP itself contains a lot of useful information regarding the request, including IRP type and the user buffer for the operation. This in turn facilitates processing of the user buffer, for example as used in UNVEIL [20], Redemption [24], ShieldFS [19] and Data Aware Defense [22].

However, developing a filesystem minifilter driver is non-trivial and could take a very long time. One reason for this is that the code runs in the kernel space, where seemingly minor bugs can result in system crashes leading to lengthy development and debugging times. If a developer wishes to sacrifice some flexibility but gain simplicity while monitoring kernel events, a primary alternative is Fibratus [28]. This is an open-source Python tool that allows the user to capture, log and process kernel events including filesystem I/O, network activity and registry activity. One example of sacrificed flexibility is that although Fibratus can filter individual filesystem I/O requests, not all of the information provided by a filesystem minifilter driver is available with Fibratus. Most notably, access to the user buffer is not provided, making it difficult to perform processing on individual filesystem writes.

The kernel space data sources include *Network*, *Registry*, *Firmware* and *Filesystem* events. Monitoring network events may reveal connections to *Command & Control Servers*, intercepted network packets could leak information such as encryption keys, and logs could reveal behaviour that is different to baseline activity. As an example, [29] and [13] detect ransomware that uses domain generation algorithms (DGAs) by monitoring DNS traffic to apply Markov Chains and behavioural-based detection features.

Monitoring changes to the registry could also be useful to detect any unexpected modifications by a malicious process such as disabling an anti-ransomware solution at start-up. Sgandurra et al. [30] uses registry key operations (along with API calls and filesystem events) as a feature for a machine learning-based approach to detect ransomware. Firmware modifications can also be used as a data source, such as in [31]. Using firmware allows access to data that doesn't exist in the operating system layer, for example whether or not filesystem writes are made to the same block of memory. As seen consistently throughout the state-of-the-art, monitoring filesystem events not only allows the analysis of I/O traces but can also potentially enable access to the user buffer itself for data processing. Finally, monitoring system events (for example process activity) could help to uncover anomalous system behaviour.

Within user space, RAPPER uses Hardware Performance Counters (HPCs) as a data source for detecting ransomware [23]. It recognises anomalous system behaviour through *System/API calls* on Linux. *Visual Output* (i.e. changes to the GUI of a system that are visible to the user) can also be used to aid in ransomware detection and classification. For example, UNVEIL uses this approach by analysing screenshots of the ransom notice with OCR and image processing.

Another approach, as seen with ShieldFS, is to analyse a process' memory for cryptographic primitives and key-related material. The authors explain that a key schedule is part of many symmetric encryption algorithms, and that this is often pre-computed and stored in the process' memory. The authors run the key-schedule algorithm and check a process' memory to see if the same values are found. This also relates to exploiting ransomware by targeting the Crypto System used to carry out encryption. Other examples of this are PayBreak [32] and UShallNotPass [33], which target cryptographic libraries that ransomware often uses. These tools implement hooking in order to intercept crypto-related API calls as a data source for their anti-ransomware methods (see Sect. 5).

Processing. In order to detect a ransomware attack, it is necessary to process the raw data in some way. This step may be as simple as *monitoring* a given data source or something more complex such as feature extraction before machine learning. *Hashing* refers to taking a malicious binary and applying a hashing algorithm to its contents, such as SHA-3. This approach is a common strategy used by antivirus vendors in order to detect and classify malware in general [34], although its usefulness in the context of ransomware is somewhat limited, in part due to the copy-cat nature of ransomware and the existence of RaaS. However, hashing has cleverly been used in the anti-ransomware domain on numerous occasions. For example, PayBreak uses a 32-byte *fuzzy function signature* in order to identify the usage of statically-linked cryptographic libraries, and CryptoDrop uses *Similarity-Preserving* hashes to quantify the difference between a file and its (possibly) encrypted version.

Another approach is to implement a score that represents the overall 'malice' of a given process, for example as implemented in Redemption and CryptoDrop. The idea here is that some indicators of ransomware behaviour can be well defined (for example how a process changes file extensions after encryption), and then applications can be monitored for occurrences of these indicators. When one such event happens, the *malice score* for the process is incremented until a pre-computed threshold is reached. At this point, the system would report that the process is likely to be ransomware and act accordingly.

Another fairly popular approach to detecting ransomware is to make use of *statistical tests*. The rationale is that properly implemented crypto-ransomware should write (encrypted) data that is effectively random. It is therefore possible to make use of lightweight, tried-and-tested statistical tests to detect the presence of randomness, and by extension, ransomware. There are several occurrences in the literature of anti-ransomware tools making use of entropy computations to help in detecting ransomware. This is often calculated over the user buffer of write requests, such as in ShieldFS. However, and as stated in [22], one weakness of the entropy test in this context is that it has difficulties distinguishing between encrypted and highly compressed data, possibly leading to many false positives if a user compresses their data or deals with compressed formats such as mp3 or jpeg. To address this issue, Data Aware Defense uses a Chi-Square test for randomness, which can distinguish between encryption and compression better.

Machine Learning. Both the raw data sources and any output computed by the processing techniques can be used as training and testing data for machine learning algorithms. A very relevant example of the use of machine learning to detect ransomware is ShieldFS. It uses a *Random Forest* algorithm to distinguish between malicious and benign system behaviour from a filesystem perspective. Examples of the features used to train this classifier include the number of files written and read and the average entropy of filesystem writes, all within a given interval. These features are derived from logs of billions of IRPs.

Another machine learning approach is the use of a neural network to classify ransomware behaviour. Whilst this often results in longer training times and produces a classifier that is difficult for humans to interpret [35], it may lead to a higher accuracy which could be crucial for end-point ransomware protection.

Actions. In order to develop a tool capable of stopping a ransomware attack, some action needs to be taken after it is decided such an attack is in progress. The most common approach is to attempt to *Kill or Block the Process or Thread* that has been classified as malicious, such as with Data Aware Defense [22]. Another potential approach could be to place it under *Surveillance*. The idea is that all processes could be monitored with quite general indicators of ransomware. If a process' behaviour begins to look malicious, the process could be placed under surveillance, i.e. more indicators of ransomware are used and more resources are devoted to its analysis. This provides the benefit of accurate decision making based on an increasing number of indicators, without the overhead of every indicator being used on every process. A similar technique is used in RAPPER.

Additionally, it is common to include some sort of *User Notification* to ensure that the decision cast by the anti-ransomware tool is sensible in a given context. For example, a user may intentionally encrypt their data, at which point some of these tools may incorrectly classify this behaviour as malicious. A notification would allow the user to continue the benign operation, or confirm the killing of a ransomware related process.

4.2 Recovery

Our analysis has shown that anti-ransomware techniques have focused on detection rather than recovery. Still, researchers are developing clever ways of recovering from ransomware attacks. Ideally, this enables the victim to revert their system to a point in time before the ransomware attack happened, mitigating the effects of the attack.

We take a similar approach in classifying recovery techniques. That is, we notice that some *Data Source* is required in order to begin recovery. This data could be, for example, some kind of backup, or access to API calls. Depending on the data source in use, some *processing* may be required before recovery is possible. After that the recovery *actions* can take place, typically via file restoration as in ShieldFS, or decryption as in PayBreak.

Data Sources. A logical way of recovering from a ransomware attack involves the use of some kind of *backup*. In the context of anti-ransomware tools, the meaning of a backup is slightly different. In the literature, anti-ransomware tools that use a backup tend to implement their own 'short-term' approach.

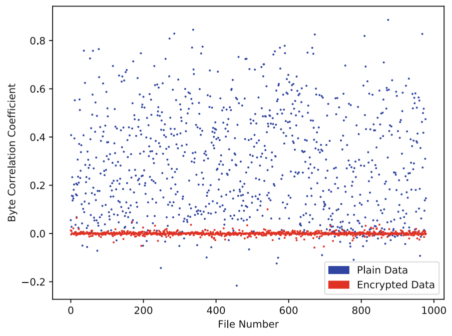
For example, ShieldFS implements a copy-on-write system that essentially creates a short-term backup of a file whenever it is written to or deleted by a process for the first time. This is achieved using the *I/O Interception* capabilities of Windows Filesystem Minifilter Drivers mentioned in Sect. 4.1. If the process is eventually classified as ransomware, the copied version of the file can be recovered. Otherwise, if sufficient time passes, the backup can be cleared.

Redemption implements a similar approach in that a write or delete will result in a copy of the file, but subsequent I/O requests to the original file will be redirected to the copy. Changes to this file are periodically written to disk unless the process is classified as ransomware. Additionally, it may be possible to implement some kind of *Cache* or *Buffer* where potential changes to the filesystem are stored until a final decision has been made as to whether or not the changes are malicious.

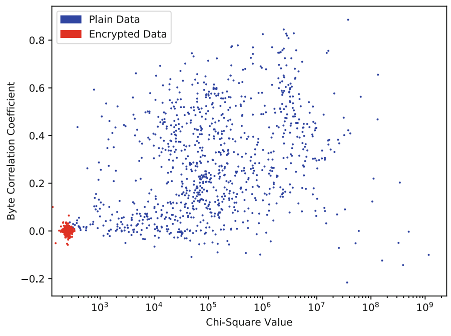
Another strategy that can aid with recovery, as explored by PayBreak, is *API Hooking*. This consists of function hooks to crypto-related libraries. PayBreak uses this technique to gather information regarding the encryption used by the ransomware, for example its symmetric key, initialisation vector and cipher mode. This is implemented using Microsoft's Detours package [36]. This information is aggregated and stored in an append-only vault, protected with administrator privileges. After a ransomware infection completes, the user is then able to activate the PayBreak recovery process at which point the collected encryption algorithm information is used with every encrypted file until successful decryption is achieved.

Processing. Processing may or may not be required, depending on the data source used for recovery. PayBreak presents an example of processing: The raw information collected from API hooking requires aggregating and storing, known as a *Key Escrow* mechanism, before being used to decrypt the files. Other examples of data processing include SSD-Insider's use of *Delayed Deletion* in order to prevent ransomware modifications being written to disk [31], and ShieldFS's use of an IRP transaction log in order to identify exactly which files were affected by a ransomware attack and need to be restored [19].

Actions. One of two major actions can be taken in order to complete the recovery process: *Restoration* or *Decryption*. As shown above, PayBreak takes



(a) Serial Byte Correlation Coefficients



(b) The Addition of Chi-Square

Fig. 2. Using serial byte correlation for ransomware detection

the decryption approach, i.e. the damage caused by ransomware is reversed via the decryption of the files affected. ShieldFS and Redemption, on the other hand, achieve recovery using restoration, i.e. the damage is reversed by restoring the unmodified versions of the affected files.

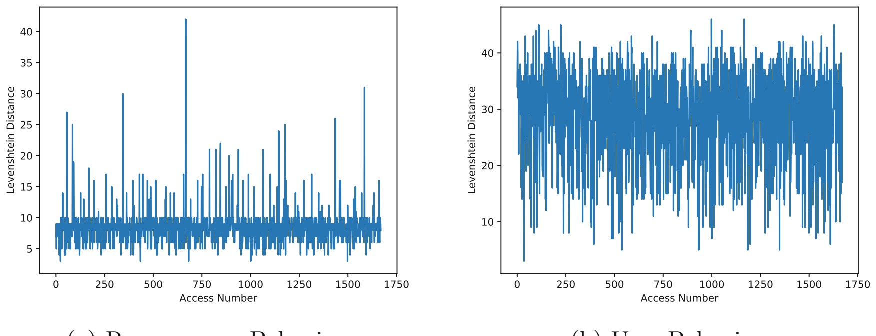
4.3 Novel Detection Techniques

Below we propose two novel indicators that show potential in detecting ransomware that – to the best of our knowledge – have not been used this way. We leave further implementation and testing of these techniques as future work.

Serial Byte Correlation Coefficient. The first is the use of the *serial correlation coefficient*, a lightweight statistical test that looks at the relationship between consecutive numbers. We look at the correlation between bytes written to a file, expecting a low value for encrypted files.

Figure 2 shows the results of experiments relating to serial byte correlation. In Fig. 2a, the serial byte correlation coefficients of 979 files from the Govdocs corpus [37] were calculated before and after encryption. A clear trend towards zero is shown for the encrypted versions of the files. Figure 2b shows values of chi-square calculated alongside byte correlation over the same data, highlighting a cluster representing random data (in this case, encrypted data) when these indicators are combined.

Edit Distance of File Paths. We also propose the use of the *edit distance* of the file path interacted with by a process. As shown by the literature [15], ransomware performs bulk encryption iteratively across files so for a given directory, we would expect to see several consecutive writes whose file paths have minimal edit distance. This is because the only part of the path that should change is the file name (and extension) itself – the bulk of the path should



(a) Ransomware Behaviour (b) User Behaviour

Fig. 3. Edit distances of file paths from filesystem accesses representing ransomware and user behaviour

remain the same until another directory is accessed. We would therefore expect ransomware to make several writes whose file paths have a very low edit distance with intermittent occurrences of high edit distances.

Figure 3 shows the differences in *Levenshtein* distance of file paths generated by iterative (Fig. 3a) and random (Fig. 3b) filesystem accesses. This quantifies the difference between given strings, or in this context, the number of edits required to get from one string to another. In order to represent the filesystem traversal of ransomware as generally as possible, we generated filesystem access requests based on the three main types of ransomware traversal reported in [15], namely depth-first with encryption starting at leaves, depth-first with encryption starting at the root, and extension-based.

Figure 3a shows the results of depth-first traversal with encryption starting at the leaves. The other behaviours generated similar patterns, although they were slightly less noticeable in the case of extension-based traversal, as there is often no guarantee that a directory will contain multiple files of the same type. Figure 3b was generated by randomising access requests to represent the unpredictability of humans, although we plan to improve this by collecting data based on real human activity.

5 Analysis and Evaluation

We mapped existing anti-ransomware tools onto our proposed roadmap, accompanied by relevant data sets and information regarding each tool's accuracy as reported by the tool's authors.

5.1 Observations

We believe that our roadmap provides a classification scheme and a clear map of the current ways ransomware is being fought, which is also expandable to cover strategies targeted at other platforms such as Android. It also highlights gaps in existing techniques that could lead to new ideas and techniques.

Table 1 provides a global view of how the anti-ransomware tools we have analysed fit into the landscape. The values shown in the blue row represent the popularity of individual techniques within the literature, whereas the values in the blue column represent how many individual features a given tool in the literature actually makes use of. Immediately noticeable is the obvious preference for detection techniques compared to recovery techniques. There is also a clear preference towards some form of monitoring, for example of the filesystem. As well as this, it is interesting to see that some reportedly promising approaches – e.g. the use of a malice score – have not received much attention in the literature.

5.2 Accuracy

Table 2 provides a comparison of current anti-ransomware tools in terms of their accuracy (i.e. their ability to successfully detect ransomware). We would like to

Table 1. Matrix of anti-ransomware tools in the landscape

	Detection																Recovery								
	Data Sources		Machine Learning					Processing					Actions					Data Sources				Proc.		Actions		
	Kernel Space	User Space	Regularized Logistic Regression	Decision Tree	Random Forest	Artificial Neural Network	Bayesian belief network	Hashing	Malice Score	Monitoring	Statistical Features	Edit Distance	Surveillance	Kill Thread/Process	Block Thread/Process	Lockdown	Notifying User	Backup	I/O Interception	Cache/Buffer	API Hooking	Key Escrow	Delayed Deletion	Restoration	Decryption	Total
UNVEIL [20]	FS	VO	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
CryptoDrop [15]	FS	-	0	0	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	8
2entFOX [29]	R FS	AC CS CE	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
Connection Monitor [13]	-	N	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	4
EldeRan [30]	R FS	AC	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
PayBreak [32]	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	3
Data Aware Defense [22]	FS	-	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4
ShieldFS [19]	FS	M CS	0	0	1	0	0	0	0	1	1	0	1	1	0	0	0	1	1	0	0	0	0	1	0	11
Redemption [24]	FS	-	0	0	0	0	0	0	1	1	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	9
UShallNotPass [33]	-	CS	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	5
RAPPER [23]	-	AC	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	5
R-Killer [38]	FS	N	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	6
SSD-Insider [31]	F	-	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	1	1	0	8
R-Locker [39]	-	HT	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	4
Honeypot [40]	-	HT	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	5
Total			1	1	1	2	1	1	2	12	7	0	2	7	2	3	9	2	3	1	1	1	1	3	1	

stress that the figures presented here are *as reported by the respective authors of the tools*. We did not have access to most of these tools, meaning we were unable to perform a fair comparison using a consistent and well designed dataset. Therefore, we leave judgment of the capabilities of the current landscape to the reader.

We also notice the reportedly high detection rates across all tools implementing filesystem activity monitoring. One such case is Redemption, purportedly achieving a detection rate of 100% and a false positive rate of 0.5% over 1,174 samples across 29 families. These results are clearly very promising. We believe that, for the task of defeating ransomware, maximising true positive rate is more important than minimising false positive rate. From the perspective of a user, a false positive (i.e. a benign process incorrectly classified as malicious) is arguably an annoyance, whereas a false negative (i.e. ransomware remaining undetected) could have catastrophic results. However, we do not disregard the importance of a low false positive rate because a user who is constantly confronted with false

Table 2. Reported anti-ransomware results

Anti-ransomware tool	Source Code	Runnable		Dataset	Ransomware		Reported results	
	Available	Free	Paid	Available	Families	Samples	Detection rate	False positive
UNVEIL [20]	✗	✗	✗	✗	N/A	3156	93.3%	0%
CryptoDrop [15]	✗	✗	✓	✗	14	492	100%	N/A
2entFOX [29]	✗	✗	✗	✗	N/A	8	87.5%	N/A
Connection-Monitor & Connection-Breaker [13]	✗	✗	✗	✗	N/A	20	100%	N/A
EldeRan [30]	✗	✗	✗	✗	11	582	$96.34 \pm 2.1\%$	$1.61 \pm 0.8\%$
PayBreak [32]	✓	✗	✗	✗	20	107	N/A	N/A
Data Aware Defense [22]	✗	✓	✗	✓	20+	798	99.37%	0.05%
ShieldFS [19]	✗	✗	✓	✓	5	383	99.74–100%	0–0.208%
Redemption [24]	✗	✗	✗	✗	29	1174	100%	0.5%
UShallNotPass [33]	✗	✗	✗	✗	N/A	524	94%	N/A
RAPPER [23]	✗	✗	✗	✗	1	1	100%	≈0%
R-Killer [38]	✗	✗	✗	✗	13	50	96%	N/A
SSD-Insider [31]	✗	✗	✗	✗	2	2	100%	5%
R-Locker [39]	✗	✗	✗	✗	2	2	100%	N/A
Honeypot [40]	✗	✗	✗	✗	N/A	N/A	N/A	N/A

positives is likely to give up on using the tool or not take appropriate action when notified about a real attack.

The authors of Data Aware Defense shift their focus to minimising system overhead, and report success in doing so ("by a factor of a few hundreds" compared with the overhead of other anti-ransomware tools [22]). However, they caution that this comparison was made without knowing the testing procedure of other tools. This shows great promise, particularly when coupled with the tool's high detection rate (99.37% over 798 samples, across more than 20 families). We believe that system overhead is a frequently forgotten but critical feature of these anti-ransomware solutions that deserves much more attention. For a user to happily use one of these tools, not only must it successfully achieve its goal of protecting them from a ransomware attack, but also their normal interactions with the system should not be significantly impacted.

The approach taken by Palisse et al. [22] in conducting a benchmarking exercise using standard third party tools is a step in the right direction. The tools that they used are CrystalDiskMark (https://crystalmark.info/en/software/crystaldiskmark/), Geekbench 4 (https://www.geekbench.com), and PCMark 8 (https://benchmarks.ul.com/pcmark8). This allows researchers to evaluate their solutions against others using the same criteria. In Sect. 5.3, we discuss how a universal testing platform could be created for evaluating anti-ransomware tools, both in terms of their accuracy and system overhead. We expect that – as ransomware detection and recovery tools become more refined – there will be a shift towards overhead minimisation. In turn, it will result in tools that are faster and more suitable for real-time end-point protection.

5.3 Limitations and Future Work

The main limitation with our analysis is our focus on PC-based anti-ransomware techniques developed by the academic and open-source community, despite the existence of tools such as Heldroid. Antivirus vendors also develop anti-ransomware tools [41,42], but we found academic and open-source tools to be more accessible, for example due to the provision of implementation details. Future work may be to expand this overview with both antivirus vendor tools and non-PC based tools to give a better overview of the anti-ransomware landscape. We are particularly interested to see any commonalities between academic and antivirus vendor techniques provide greater insight into popular and underdeveloped areas. We believe it would also be interesting to see how techniques from both communities evolve over time. It would be fascinating to see how advances from one community inspire further advances in the other, leading to a cycle of continuous improvement in anti-ransomware techniques.

As mentioned in Sect. 5.2, we note that the performance and overhead statistics we have provided are as self-reported by the authors of the respective tools themselves. Therefore we do not believe it possible to conduct a fair comparison of the effectiveness of each technique. Another area of future work would be to develop a universal testing platform such that each of the tools can be evaluated in isolation using the same data sets and be fairly and transparently evaluated on the same criteria. It could be possible to develop such a platform using virtual machines (VMs). Snapshots could be taken of VMs in their fresh states (i.e. a clean installation of the target OS) and then the VMs could be configured with the anti-ransomware tool to test. Additionally, it could be possible to automate the entire process, taking inspiration from the automated malware analysis platform developed by the authors of [22].

6 Conclusion

In this work, we have presented a clear and simple roadmap of the current academic and open-source anti-ransomware landscape. This encompasses the current techniques being used to detect and recover from ransomware attacks, from the point of view of *Data Sources*, *Processing* and *Actions*. We used these classifications to provide both a consistent terminology for researchers in the area, as well as the ability to accommodate new techniques in the future. On top of that, we proposed, implemented and tested two new techniques for ransomware detection, using serial byte correlation and edit distance.

We also examined how existing anti-ransomware tools (including our proposed techniques) fit into the landscape, noticing a current preference towards filesystem activity monitoring for detection. We also provided a single point of reference comparing reported results of current anti-ransomware tools as well as their dataset sizes. We hope this information provides useful insights into current and future trends in fighting ransomware. We also believe that a clear roadmap of the landscape, along with a consistent terminology, will help to simplify and organise the development of improved future anti-ransomware techniques.

This work has been carried out to the best of our ability with limited access to the anti-ransomware tools themselves. In the interests of scientific reproducibility, we are happy to provide all of the material required to repeat the experiments discussed in this work.

Acknowledgement. Part of the work presented in this paper has been funded by the UK Engineering and Physical Sciences Research Council (EPSRC) Project EP/P011772/1 on the EconoMical, PsycHologicAl and Societal Impact of RanSomware (EMPHASIS).

References

1. Varonis: A brief history of ransomware (2016) https://www.varonis.com/blog/a-brief-history-of-ransomware/
2. Young, A., Yung, M.: Cryptovirology: extortion-based security threats and countermeasures. In: Proceedings 1996 IEEE Symposium on Security and Privacy, pp. 129–140, May 1996
3. Arsene, L., Gheorghe, A.: Ransomware, a victims perspective (2016). http://www.bitdefender.com/media/materials/white-papers/en/Bitdefender_Ransomware_A_Victim_Perspective.pdf
4. Dunn, J.E.: Sophoslabs (2018). https://nakedsecurity.sophos.com/2018/11/14/targeted-ransomware-attacks-sophoslabs-2019-threat-report/
5. Cartwright, E., Hernandez Castro, J., Cartwright, A.: To pay or not: game theoretic models of ransomware. J. Cybersecur. **5**(1) (2019). https://doi.org/10.1093/cybsec/tyz009
6. Hernandez-Castro, J., et al.: Economic analysis of ransomware. CoRR, abs/1703.06660 (2017). http://arxiv.org/abs/1703.06660
7. Kevin Savage, H.L., Coogan, P.: The evolution of ransomware, August 2015. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf
8. Hart, N.: The New Economy (2018). https://www.theneweconomy.com/technology/raas-satans-business-model
9. BBC News: Huge aluminium plants hit by 'severe' ransomware attack (2019). https://www.bbc.co.uk/news/technology-47624207
10. No More Ransom (2019). https://www.nomoreransom.org
11. Trend Micro: Best practices: Ransomware (2017). https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/best-practices-ransomware
12. Al-rimy, B.A.S., et al.: Ransomware threat success factors, taxonomy, and countermeasures: a survey and research directions. Comput. Secur. **74**, 144–166 (2018)
13. Ahmadian, M.M., Shahriari, H.R., Ghaffarian, S.M.: Connection-monitor connection-breaker: a novel approach for prevention and detection of high survivable ransomwares. In: 2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), pp. 79–84, September 2015
14. Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., Kirda, E.: Cutting the gordian knot: a look under the hood of ransomware attacks. In: Almgren, M., Gulisano, V., Maggi, F. (eds.) DIMVA 2015. LNCS, vol. 9148, pp. 3–24. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20550-2_1

15. Scaife, N., et al.: Cryptolock (and drop it): stopping ransomware attacks on user data. In: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp. 303–312, June 2016
16. Gazet, A.: Comparative analysis of various ransomware virii. J. Comput. Virol. **6**(1), 77–90 (2010)
17. Mercaldo, F., et al.: Ransomware inside out. In: 2016 11th International Conference on Availability, Reliability and Security (ARES), pp. 628–637, August 2016
18. Song, S., Kim, B., Lee, S.: The effective ransomware prevention technique using process monitoring on Android platform. Mob. Inf. Syst. **2016** (2016). Article ID 2946735, 9 p. https://doi.org/10.1155/2016/2946735
19. Continella, A., et al.: ShieldFS: a self-healing, ransomware-aware filesystem. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016, pp. 336–347. ACM, New York (2016)
20. Kharraz, A., et al.: UNVEIL: a large-scale, automated approach to detecting ransomware. In: 25th USENIX Security Symposium (USENIXSecurity 16), pp. 757–772. USENIX (2016)
21. Andronio, N., Zanero, S., Maggi, F.: HelDroid: dissecting and detecting mobile ransomware. In: Bos, H., Monrose, F., Blanc, G. (eds.) RAID 2015. LNCS, vol. 9404, pp. 382–404. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26362-5_18
22. Palisse, A., Durand, A., Le Bouder, H., Le Guernic, C., Lanet, J.-L.: Data Aware Defense (DaD): towards a generic and practical ransomware countermeasure. In: Lipmaa, H., Mitrokotsa, A., Matulevičius, R. (eds.) NordSec 2017. LNCS, vol. 10674, pp. 192–208. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70290-2_12
23. Alam, M., et al.: RAPPER: ransomware prevention via performance counters. abs/1802.03909 (2018). http://arxiv.org/abs/1802.03909
24. Kharraz, A., Kirda, E.: Redemption: real-time protection against ransomware at end-hosts. In: Dacier, M., Bailey, M., Polychronakis, M., Antonakakis, M. (eds.) RAID 2017. LNCS, vol. 10453, pp. 98–119. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66332-6_5
25. Greenberg, A.: The untold story of NotPetya, the most devastating cyberattack in history (2018). https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/
26. Hull, G., John, H., Arief, B.: Ransomware deployment methods and analysis: views from a predictive model and human responses. Crime Sci. **8**(1) (2019). https://doi.org/10.1186/s40163-019-0097-9
27. Microsoft: File system minifilter drivers - windows drivers — microsoft docs (2017). https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/file-system-minifilter-drivers
28. Sabić, N.: Fibratus (2016). https://github.com/rabbitstack
29. Ahmadian, M.M., Shahriari, H.R.: 2entFOX: a framework for high survivable ransomwares detection. In: 2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), pp. 79–84, September 2016
30. Sgandurra, D., et al.: Automated dynamic analysis of ransomware: benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020 (2016)
31. Baek, S., et al.: SSD-insider: internal defense of solid-state drive against ransomware with perfect data recovery. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 875–884, July 2018

32. Kolodenker, E., et al.: Paybreak: defense against cryptographic ransomware. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 599–611. ACM (2017)
33. Genç, Z.A., Lenzini, G., Ryan, P.Y.A.: No random, no ransom: a key to stop cryptographic ransomware. In: Giuffrida, C., Bardin, S., Blanc, G. (eds.) DIMVA 2018. LNCS, vol. 10885, pp. 234–255. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93411-2_11
34. Virus Total: Virustotal-free online virus, malware and URL scanner (2012). https://www.virustotal.com/en
35. DTREG: Decision trees compared to regression and neural networks (2019). https://www.dtreg.com/methodology/view/decision-trees-compared-to-regression-and-neural-networks
36. Microsoft: Detours (2016). https://github.com/Microsoft/
37. Digital Corpora (2018). https://digitalcorpora.org
38. Lokuketagoda, B., et al.: R-killer: an email based ransomware protection tool. In: 2018 13th International Conference on Computer Science Education (ICCSE), pp. 1–7, August 2018
39. Gómez-Hernández, J., et al.: R-locker: thwarting ransomware action through a honeyfile-based approach. Comput. Secur. **73**, 389–398 (2018)
40. Moore, C.: Detecting ransomware with honeypot techniques. In: 2016 Cybersecurity and Cyberforensics Conference (CCC), pp. 77–81, August 2016
41. BitDefender (2019). https://www.bitdefender.com/business/cyber-threats-solutions/anti-ransomware.html
42. MalwareBytes (2019). https://www.malwarebytes.com/business/solutions/ransomware/

An Experimental Analysis of Cryptojacking Attacks

Per Håkon Meland[1,2](✉), Bent Heier Johansen[2], and Guttorm Sindre[2]

[1] SINTEF Digital, Trondheim, Norway
per.h.meland@sintef.no
[2] Norwegian University of Science and Technology, Trondheim, Norway
{per.hakon.meland,guttorm.sindre}@ntnu.no, bent.heier@gmail.com
https://www.sintef.com, https://www.ntnu.edu

Abstract. Cryptojacking is the illicit exploitation of Internet users' bandwidth and processing power to mine cryptocurrencies. This paper presents an experimental analysis of how different types of cryptojacking attacks impact a selection of consumer-grade devices, and the perceived annoyance by the user. This is seen in relation to the expected cost and revenue the attacker would expect. The results show that a well-configured cryptojacking attack does not significantly harm its victims, hence can be very difficult to detect, and even aware users might not bother getting rid of the infection. The costs and risk associated with performing cryptojacking are low, but the attacker would rely on a pool of infected devices over a prolonged period of time in order to make any significant revenue. The main cost is therefore the opportunity cost, as there are more profitable ways to abuse compromised systems due to the general decline in cryptocurrency values. Though the heyday of cryptojacking has gone by, several adversaries are likely to have made quite a profit from it. It can therefore emerge as a serious threat again due to market externalities.

Keywords: Cryptojacking · Cryptomining · Drive-by mining · Monero · Blockchain · Malware · Experiment · Economy

1 Introduction

Cryptojacking is one of the youngest members in the family of cryptocurrency related crimes, including blatant theft, illegal trading, money laundering, extortion and ransomware among others. Since the investment costs of hardware and electricity in most cases exceed the expected profit from mining cryptocurrencies, the goal with cryptojacking is to illicitly exploit Internet users' bandwidth and processing power to mine on behalf of the attacker. In contrast to many other types of attacks, cryptojacking is not about stealing or altering data, nor does it want to interrupt the victims workflow or operations. Instead it wants to stay hidden and extract as many CPU cycles as possible. In 2018, Europol [17] proclaimed that the industry reported an explosion in the volume of illicit cryptomining and in the latter part of 2017, it overshadowed almost all other malware threats. However, in a more recent report from 2019 [40], Symantec finds that cryptojacking dropped by 52% between January and December 2018, and that the declining trend is continuing.

In order to explain some of the reasons why cryptojacking promptly declined as a cyber threat, this paper presents an experimental analysis of cryptojacking impacts using a selection of six types of consumer-grade devices. This is seen in relation to the expected cost and profit the attacker would expect. The goal of the experiment has been to answer the following research questions:

1. How is performance on different types of devices affected by cryptojacking measured objectively and perceived subjectively?
2. What are the expected revenues and costs for the attacker based on the targeted devices?

The outline of this paper is as follows: Sect. 2 explains the basics of cryptojacking attacks and cryptocurrencies typically associated with them. In Sect. 3 we present the experiment setup and measurement types. Section 4 presents the results from the experiment in terms of mining efficiency and how the devices are degraded seen objectively and subjectively. Section 5 discusses mining times and investments compared to expected profit, as well as limitations and related work. Finally, Sect. 6 revisits the research questions and concludes the paper.

2 Background

2.1 Types of Cryptojacking Attacks

There are two main types of cryptojacking attacks; one which require a malicious payload to be installed on the user's computer and the other which runs inside the user's browser upon visiting dubious web sites. In the former case, the simplest attacks typically fool users to download and launch an executable file or open an email attachment. More advanced methods exploit unpatched vulnerabilities, often zero-days to bypass the user entirely and install the payload.

The second type is an even more subtle way of attacking. About 95% of all web sites use *JavaScript* [43], and due to its popularity JavaScript is supported by all major web browsers. JavaScript is a quite powerful programming language running inside the web browser and uses the computing power of the client, not the web server. This allows for a lot of processing power, including the power to mine cryptocurrency. Such a *drive-by download* attack [11] terminates as soon as the web page is closed, leaving no trace on the victim's computer.

The most well-known script for cryptojacking was offered by *Coinhive* [7]. It allowed web site owners to deliberately put a cryptominer on their web site, letting visitors choose to allow the use of their CPUs for mining. However the same script was also frequently injected into compromised sites [12,32]. The business model of Coinhive was to take 500 EUR for an account creation, then a 30%

share of the mining itself. The services offered by Coinhive were not nefarious or illegal, in fact, they presented themselves as an alternative to advertisement, which is one of the main sources of revenue on the Internet to day. However, Coinhive was quite controversial and received their share of criticism since their initial script did not ask web site visitors for consent, and the users did not have to upgrade to the one that required this. On February 26th 2019, the Coinhive Team announced they were shutting down their service as of March 8th 2019. They proclaimed that it was no longer profitable to keep the service operating anymore, citing that Monero had depreciated more than 85% over the last year and that the hashrate dropped over 50% after the last hard fork [8].

Coinhive accounted for 70–75% of the cryptojacking JavaScripts on the web in 2018 [31,35], and while the default setting was to use 100% of the victims CPU, researchers have found that most sites throttled themselves to use between 25% and 70% of available CPU power [16,27]. This was likely done to make the mining unnoticeable to the user. A report by Rüth et al. [35] also found that merely 10 user accounts were responsible for 80% all short links, meaning that only a handful of people were reaping the vast majority of the profits.

An alternative approach for cryptojacking is to mine using *plugins* that are used by web sites, such as *Wordpress* plugins. This will require a compromise of the browser extension itself, which is easier to detect. In the past, Wordpress had cryptomining plugins on its official plugin page, including several miners using the Coinhive script [44]. These could be included by legitimate web site owners, but they could also be deployed on compromised sites. *Browser extensions* are yet another vector for attackers.

Cryptojacking can also target smart phones and IoT devices. For the Android operating system, it is possible to download applications as *APK*-files from the Internet and install them directly without going through Google's Play Store. If the *side loading* setting is not set to off, cryptomining apps like *HiddenMiner* [45] can take advantage of the device. The auto update feature can also be exploited to install a cryptominer. It should be noted that Google have recently removed all cryptomining apps from the Google Play Store. Apple's iOS has been less susceptible to these kinds of attack due to a stricter lock-down policy. However, there have been incidents where apps suddenly begun mining cryptocurrency, such as the *Calendar 2* app [24]. Apple has also proclaimed that they do not allow cryptominers in their App Store [4], but mining can still be done using developer accounts or a jailbroken device.

Luckily the security industry has developed many techniques to prevent cryptojacking [38]. For native miners, all the same procedures that prevent other kinds of malware will be effective. For instance, anti-virus programs have caught up and can detect the well-known cryptominers [14,18,20]. To protect against web miners there exist a lot of options as well, such as *browser extensions*, specialized *addons* and general purpose *ad-blockers*.

2.2 Coins Suited for Cryptojacking

The first mainstream and most popular cryptocurrency, *Bitcoin*, was created to establish a decentralized global currency [28]. While Bitcoin in theory is anonymous, linking an account to a person is considered manageable when the coins at some point are exchanged or used to buy items. To preserve the integrity of the blockchain, all Bitcoin transactions and associated wallets are public. This means that if a person is linked to a wallet, all previous transactions can be traced back as well.

Monero [26] is a cryptocurrency based on Bytecoin. Bytecoin was abandoned when the community found out that its creators had mined about 80% of the supply for themselves, but the technology was sound, so Monero rose from the ashes. Monero uses an algorithm called *CryptoNote*, which is virtually untraceable and unlinkable [37]. This is a desirable feature when you are exploiting somebody else's hardware. Monero is currently only on the 10th place among cryptocurrencies when in comes to market capitalization [9], however it is a very popular payment option among Dark Net marketplaces trading illegal goods and services.

In cryptomining everyone that mines is competing to solve the next block and get the next payout. Bitcoin and similar technologies use primarily raw computing power and can be effectively done in parallel. This makes expensive *High Performance Computers* (HPC) desirable targets for native Bitcoin mining. However, these machines tend to be well protected and not easy to infect with native cryptojacking attacks.

CryptoNote is less CPU intensive, but requires a relatively large amount of memory (CPU-cache or RAM) instead. Compared to Bitcoin, the benefits of using large computing clusters, GPUs and ASICs over regular CPUs are severely diminished. This means that average consumer-grade hardware has a decent chance of solving the puzzle and get the payout. This in turn makes Monero an attractive currency to mine when someone has access to a large number of regular and cheap devices, such as laptops, IoT-devices and smart phones. These devices exist in enormous quantities around the world with limited protection, hence very suitable targets for cryptojackers seeking Monero.

3 Method

For our cryptojacking experiment we decided to focus on Monero mining and a selection of consumer-grade devices typically found in homes and work places. The goals were to understand how Monero performs under different configurations, how efficient web mining is in comparison to native mining, how much power is consumed and how noticeable this kind of mining would be on an infected device. The devices we included were the following:

– **NUC** (Intel NUC7i5BNK) was a tiny computer running Linux Ubuntu 18.04. It had a two core, four thread, i5 2.3 GHz CPU with a turbo mode at 3.4 GHz, 4 MB CPU cache memory and 8 GB RAM. It was released in Q1 2017 and represents low-to-medium powered computers.

- **Mac** was a mid-2014 laptop running MacOS High Sierra. It had a dual core, four thread CPU running at about 2.6 GHz with a turbo mode at 3.1 GHz. With 3 MB CPU cache memory, 8 GB RAM and no discrete graphics, it was chosen to represent laptops.
- **Chromebook** was a low powered ASUS laptop running Chrome OS with developer access. It had a 2.16 GHz dual core CPU without hyper threading, 1 MB of L2 cache and 2 GB RAM. It represents devices that do not have true access to the hardware.
- **Stationary** was a custom made desktop PC with Microsoft Windows 10, build in early 2014 with a four core, eight thread, i7 CPU running at 3.40 GHz, with turbo up to 3.90 GHz, 8 MB of L3 cache, 16 GB RAM and a discrete Nvidia GTX 760 graphics card, making it the most powerful device in the experiment.
- **Phone** was a Sony H4113 Android smart phone from 2018 with root access. It had two ARM CPUs, both dual core, four thread, one running at 2.2 GHz and one running at 1.8 GHz with 3 GB RAM. It was included to analyze web mining on phones.
- **Rpi** was a Raspberry Pi 2 Model B with a 900 MHz ARM Cortex-A7 CPU with 256 KB of L2 cache and 1 GB RAM, This was the least powerful device used it this experiment and dates back to 2015. It represents fairly advanced IoT devices and was only used for native mining.

In order to determine Monero performance, the hashrate was the main parameter. The peak and average hashrates were recorded by the mining software. The peak tells us what the device is capable of when the miner has most of the device's resources for itself, while the average hashrate tells us how much is likely to be mined when the device is used in a regular manner. The tests were performed with a varying amount of threads mining simultaneously, which allowed us to see the overhead effects as well.

For the native tests a miner called *XMR-stak* [46] was used. It runs natively on x86 versions of Linux, Windows and MacOS for both the CPU and GPU. Unfortunately, XMR-stak does not run on ARM devices, and as thus it could not be used on the Rpi. Instead another program, *cpuminer-multi* [13] was used in this case. We also employed a mining pool named supportxmr.com [39] that allowed us to extrapolate the number of required hashes for one coin of Monero without actually having to mine a whole coin. A mining pool works by connecting the resources of many miners together. When a block is solved, every member of the pool gets a share of the coinage based on the amount of work they contributed. In this way a mining pool can provide a steady and predictable income as opposed to the random nature of solo mining. The effectiveness of web mining was measured by employing several different web sites, including coinhive.com [7] (before its shutdown), coinwebmining.com [10] and minero.cc [25].

Each device ran for at least 1 hour for each configuration of native mining and for at least 10 min of web mining. Though the time intervals are somewhat short, initial testing showed that the hashrate was quite stable, so it was deemed unnecessary to prolong the experiment. The web mining gave a real time update and had far less variance than the native mining.

The power consumption was measured by the spot Watt usage when running idle and when mining under different configurations. The consumption was measured over a few minutes, this was enough time to get an estimate that could be extrapolated. For the devices that did not have batteries (NUC, Stationary and Rpi), a simple hardware power recorder was installed in the power outlet and read directly. For the Chromebook, a build-in utility was used (`chrome://power` in the URL-bar). With the Macbook we used a utility called *iStats Menu* [21], and with the Phone *Android Studio* and *Battery Historian* [3] were used.

To measure the actual impact of cryptojacking, we had one objective and one subjective approach. Objective measurements were collected with *Sysbench* [41] on the supported systems. On Android no comparable benchmarking tool to Sysbench was found and thus no benchmark data have been collected for the Phone.

For the subjective testing, a scale of annoyance was recorded for the different configurations. It ranged from *0 - not annoyed at all* to *4 - the device is practically unusable*. The devices were tested doing some common tasks such as surfing the web, streaming HD-video, using office applications and gaming.

4 Results

4.1 Relationship Between Hashrate and Power

Figure 1 shows the highest recorded hashrate and power consumption for the NUC in both native and web mining mode when varying the number of threads used.

When mining natively it peaked at 2 threads, and decreased somewhat when adding more threads, probably due to increased overhead. During web mining adding more threads seemed to work well to increase the hashrate, although the 4th threads did not add much. The power consumption was very similar between native and web mining, at about three times the power consumption when idling. Interestingly, adding more threads to mine did not increase the power consumption by a whole lot.

We saw similar trends with the other devices as well, native mining outperforms web mining by a factor between 3–8, and in most cases uses less power. With the Mac, adding a second thread does not affect the hashrate beyond the margin of error, and the third thread adds less than a 10% increase. The fourth thread does not add anything at all. The Chromebook did not mine very efficiently, but the power consumption was also quite low. Both the Rpi and Phone scaled almost linearly when adding threads for mining. With the Stationary, we noticed that the power drain during mining was almost exactly the same whether mining natively or web, but when using the GPU the power consumption went up significantly. The next noticeable thing was that the hashrate peaked at four threads during native mining, GPU or no GPU. Adding even more threads

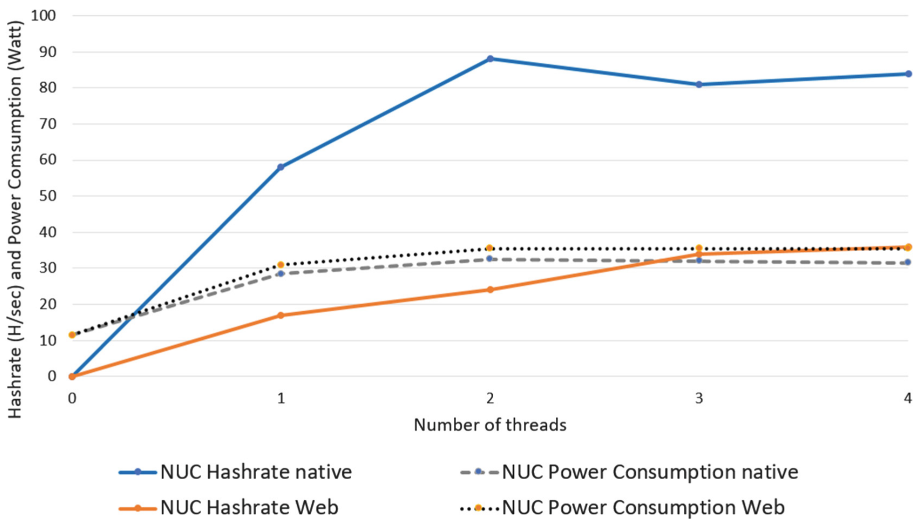


Fig. 1. Hashrate and power consumption of the NUC in native and web mining modes.

made the hashrate drop significantly. This was not true for web mining where the hashrate flattened out.

Figure 2 shows mining performance on all devices where the hashrate has been divided by the power consumption. The device that truly stands out is the Rpi at 3 threads, but this was not very efficient mining. The highest hashrate of the Rpi was 12.1, while the Stationary had 218.7 in native mode with the same number of threads.

4.2 Objective Impact on Performance and Latency

To get measurements on the impact cryptomining had on device performance, Sysbench was used both while the devices were idling and mining. Sysbench works by running a large amount of math problems by the CPU to test how many events it can process in a given time. The number of threads for both mining and performing events were varied to see how this competition for resources turned out. Figures 3 and 4 show how mining affected performance and latency for the NUC. For the Mac and Stationary the trends were the same, when mining at full speed the performance of all devices drop to about half and the latency increases dramatically.

4.3 Subjective Impact on Casual Use

While a benchmark is very useful for getting an objective measurement on how a stressed device is affected by cryptojacking, this does not necessarily tell the whole story. If the users are not bothered by cryptominers using their CPUs, they

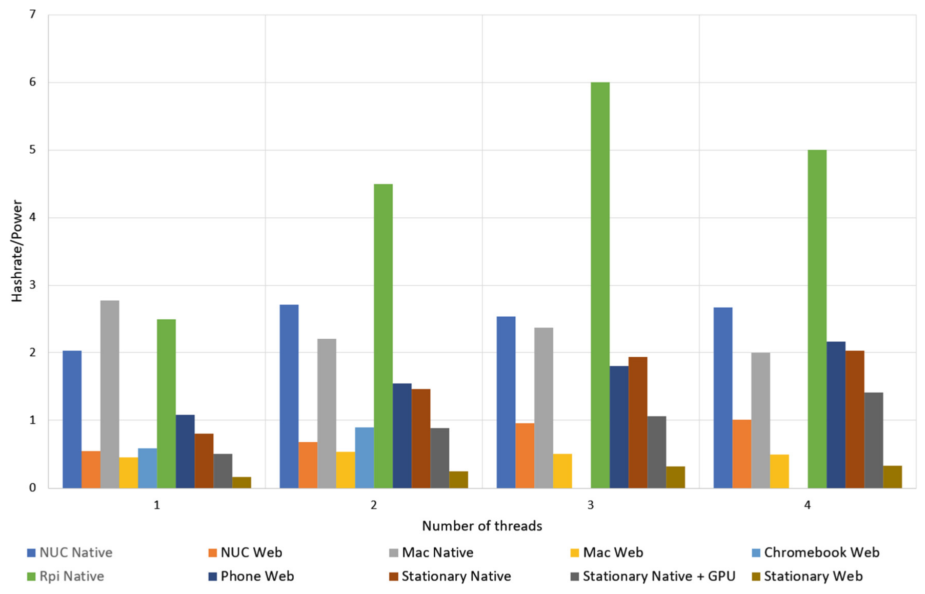


Fig. 2. Relationship between hash efficiency and power consumption (higher is better).

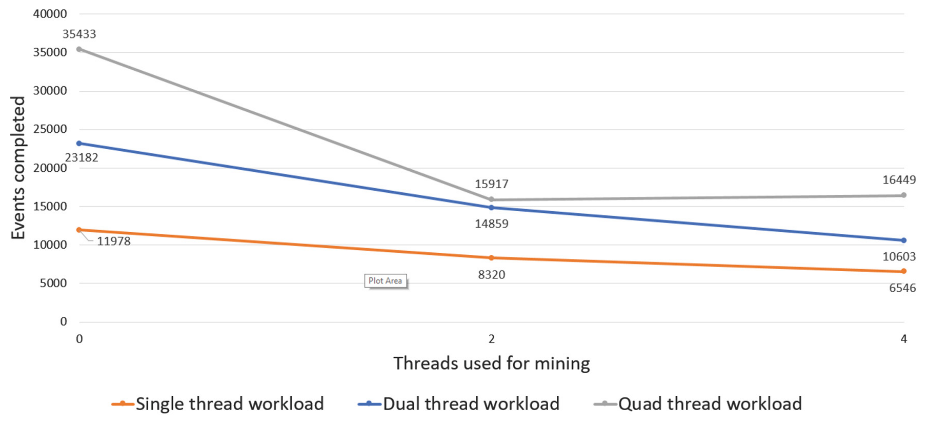


Fig. 3. NUC performance during mining (higher is better).

are less likely to do anything about it. In Fig. 5 we have recorded the annoyance level for five of the devices when they were exposed to an increasing number of threads used for mining. The Phone was omitted since multitasking (running several apps in parallel) could not be done in practice while mining. The scores were all given by a single individual (one of the authors), the results are thus highly subjective and not very reliable, but they still give an indication of how mining might impact the perceived performance of cryptojacked devices.

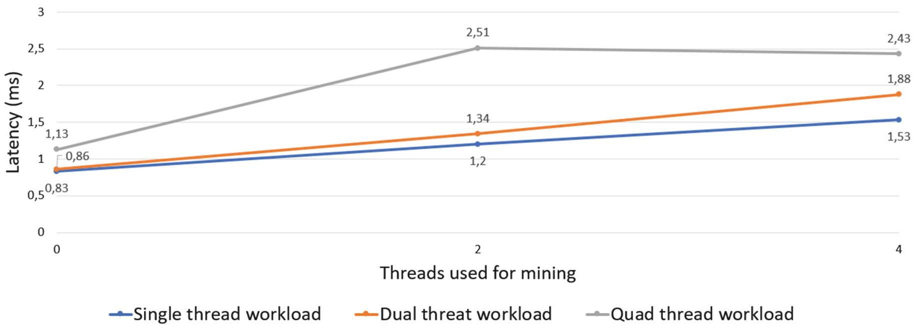


Fig. 4. NUC latency during mining (lower is better).

The stationary was tested while performing several different tasks including steaming HD video, working with office documents and playing some game (real-time and turn-based strategy games, turn-based card game, real-time fighting game). When mining using the GPU in XMR-stak and no CPU threads the graphical I/O were severely impacted, to the point of making the whole computer unusable for anything else. However, when running as many as 7 out of 8 CPU threads the impact was negligible when simultaneously streaming HD-video and playing games. When running all 8 threads the impact was noticeable, but the computer was still fully usable. Even so, the increased latency was only significantly noticeable when performing context switches, such as loading new maps in a game, starting a new video, open new documents for editing and switching between different web sites rapidly. When staying within a single application, document or map for a long time the perceived performance hit was much less noticeable. The NUC and Mac were tested in much the same way, but with fewer games. The results were similar, both devices were slower than the Stationary even with no mining, but the reduced performance was only noticeable when using all available threads for mining. Latencies in load times and context switching were somewhat more noticeable on these devices. On the Chromebook, a web miner was set up running at 100% using both available threads. At the same time full HD videos were streamed and documents were opened in the browser. While the Chromebook was slower in comparison to the other machines, there was little performance impact from the web miner. The Rpi was only tested with a web browser running in the GUI. It was very slow to begin with and the mining made it virtually unusable.

5 Discussion

5.1 Best Buck for the Bang

An important aspect of cryptomining is how long it takes to accumulate the currency. In our case we worked with Monero, and during our experiment we ran about 223,680,786 hashes that generated about 0.0125 XMR. This indicates that

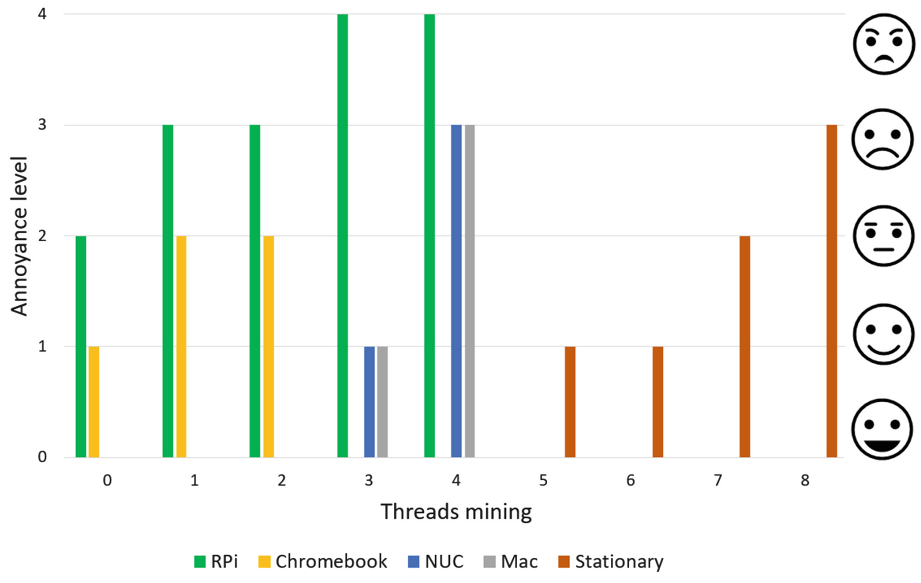


Fig. 5. Subjective experience of performance during mining.

it takes about 18 billion ($1.789\,446\,288 \times 10^{10}$) hashes to produce a single coin. Note that this estimate is subject to the ever-changing nature of Monero mining. The Monero network limits the payouts to only once every second minute, thus the more participants in the network the more hashes are required to acquire one coin. Additionally, the payouts decrease over time, effectively increasing the amount of hashes necessary to acquire one coin. Since our work was done prior to the hard fork on the 9th of March 2019, we have looked at the potential value at that time. On the 1st of March 2019, one coin had a value of about 50 USD [9]. As with most cryptocurrencies, this number fluctuates a lot. On the 7th of January 2018, Monero peaked at about 500 USD. Even so, we can use our estimates to make a comparison between the devices and give an indication of how long a miner must run to yield valuable results on a single device. This is shown in Fig. 6, where we have used the maximum recorded hashrate from our devices.

As can be seen from the figure, even greedy configurations of the script need years to mine a single coin even when running on high-end devices. In order to have any reasonable chance of making a revenue from this kind of mining, a cryptojacker would need to infect a large number of devices, preferably in native mode. For cyber criminals this means that there might be more profitable ways to use compromised devices, such as encrypting the data and demanding ransom, have the device participating in denial of service attacks or just leave it dormant until some use for it can be found.

Rational attackers will not only consider the potential revenue of cryptojacking, but also their own investments in order to perform the attack.

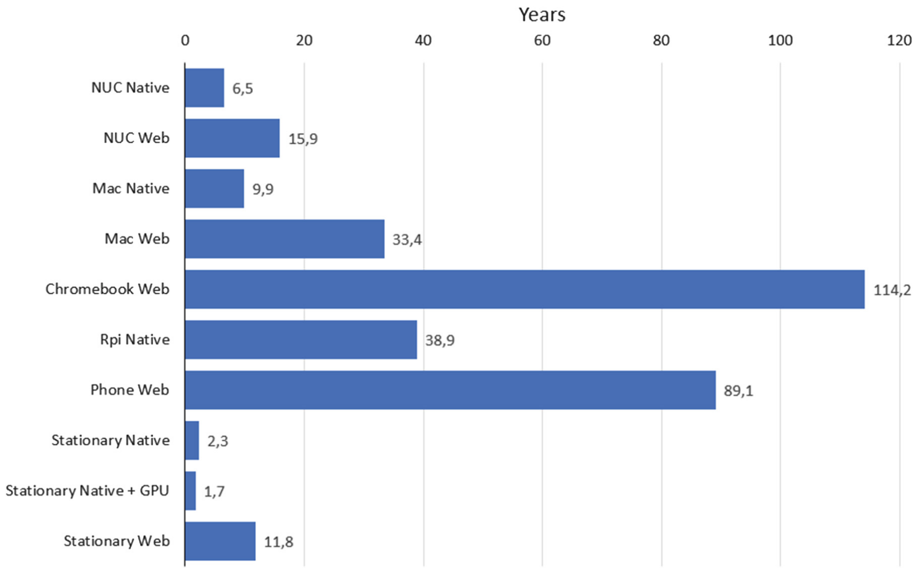


Fig. 6. Years to mine a single Monero coin (lower is better).

Since cryptomining by itself is perfectly legal, a lot of the necessary code can already be found in the public domain, thus the job of the cryptojacking developers is to weaponize the code. This includes making it run stealthily and undetected by the user and perhaps include an auto update feature. As an example, we can look at *DeepMiner*'s source code [15], which is freely available and constitutes about 1000 lines of code when excluding the cryptography itself. Assuming an investment of 18 USD per line [34], a rough estimate would be 18 000 USD for the weaponization. Alternatively, the attacker could buy off-the-shelf cryptojacking software from e.g. Dark Net markets. We have observed such items being sold for about 150 USD, although the prices very between vendors and markets. However, there is a significant risk of being scammed when purchasing items on the Dark Net, which must also be considered in this equation. Once created, the cryptojacking software must be maintained and updated, which can be even more challenging than for legitimate software. Malware usually takes advantage of some vulnerability to infect other software, but such vulnerabilities are patched regularly, so there is a limited window of opportunity. Additionally, anti-virus programs and ad-blockers will quickly be on the lookout for cryptojacking signatures and behavior. There is also a significant cost of distribution, which is difficult to put a price tag on. Often the people distributing cryptojacking attacks are not the same people that wrote the software. This was the whole business idea behind Coinhive and its affiliates. Native miners have an even higher distribution cost in order to be installed and executed, whether through social engineering, as a trojan, using an exploit or through physical access to the devices.

Some of these costs might be better measured in terms of time rather than money spent, which we often refer to as opportunity cost. Opportunity cost refers to the cost of doing one thing rather than another. Every hour, every buck and every bit of effort put into cryptojacking attacks could be used to do something else. As we have shown in Fig. 6, an attacker would need long-term infection periods on a vast number of devices just to make some small revenue, and it is therefore understandable why the heyday of cryptojacking has gone by. Having said that, the cryptocurrency market might skyrocket again, making cryptojacking a very relevant threat again.

5.2 Limitations

One obvious limitation in our experiment is the relatively small sample size of six devices, running different operating systems and hardware configurations. Also, the subjective annoyance recording could have involved more people, but it is doubtful that the results would have been very different. Additionally, it was difficult to account for other running processes even when comparing idle states with mining activities. Prolonged mining would also create a temperature increase making the different devices behave differently, but this was not something we recorded.

Monero's Cryptonote mining algorithm was using memory blocks of 2 MB at the time of our experiment, meaning that in theory each CPU or GPU thread running CryptoNote would be most efficient if they could get 2 MB of cached memory for themselves. There are now plans for Monero to switch to another proof-of-work algorithm that requires miners to dedicate over 2 GB of RAM to the process, making cryptojacking attempts harder to hide [48] and probably useless on low-end devices. It would therefore be useful to repeat the experiment as the algorithm changes to see how this affects the impact on different device types.

5.3 Related Work

Cryptojacking is a relatively new phenomenon, hence there has been limited research on this kind of threat prior to 2018. Musch et al. [27] wrote an extensive report on web-based cryptojacking this year, describing how to identify mining scripts among the Alexa [2] Top 1M web sites and expected mining revenues. They found that about 1 out of 500 web sites contained miners and that there was moderate profit to be made at that time. Tahir et al. [42] have done a later study on Alexa Top 50K web sites looking for cryptojacking, and also discovered that mining-prevention plugins often fail to detect such scripts.

Eskandari et al. [16] have analyzed the profitability of cryptojacking web sites using a real-world data set, showing that over a period of three months little revenue can be earned. They also discuss whether the web site visitors giving consent to mine have a clear mental model of what they are paying. This is supported by Carlin et al. [6], who discuss the legality of cryptomining, referring to UK legislation.

Similarly to our work, Saad et al. [36] have analyzed the impact cryptojacking has on system resources on various devices, in their case three different laptops and one smart phone, but only for web-based mining. They also examined the economic basis for cryptomining as an alternative to advertisement on web sites, and concluded that cryptomining was not a feasible alternative. In parallel to this work, Papadopoulos et al. [31] studied the profitability of in-browser mining and developed a testbench that ran on a Linux desktop. They found that advertisements were 5.5 more profitable than web-cryptomining, but that hybrid solutions would allow for maximum profits. However, on the user side the device temperature and power consumption would increase 52.8% and 2X respectively on a desktop computer.

Hong et al. [19] have done a systematic study on cryptojacking and present a detector that automatically tracks mining scripts. This detector has been applied to the Alexa Top 100K list, and they estimated a danger to more than 10M web users and extra spending of electricity that is similar to powering a city. Kharraz et al. [22] present another detector that has been applied to Alexa Top 1M and conclude that cryptojacking operations can be detected with minimal human interventions. Konoth et al. [23] did another crawl of Alexa Top 1M, but in contrast to related studies, they analyzed more than just the landing pages. They found that only 3.86% of cryptomining web sites informed their users of this activity, and that the most profitable web site was earning 17K USD a month from 29M visitors. However, the vast majority of web sites were making very little revenue from cryptomining. Pastrana and Guillermo [33] have conducted a longitudinal study where they analyze about 1M malicious miners to see where the profit goes in the underground economy. They found that at least 56M USD have gone to criminals. A broader paper on how to monetize from web attacks has been published by Nguyen et al. [29], who also suggest countermeasures to this. A paper by Norman [30] also focus on many of the same countermeasures. In a review paper by Al Hajri et al. [1] a particular warning goes to enterprises due to their broad attack surface.

Sigler [38] show the trend where web/script-based cryptojacking attacks became more favorable than the native counterpart due to their easiness. Zimba et al. [47] have proposed how digital autopsies on both native and web-based miners, as well as extortion malware, can be performed. They found that most of the scripts they analyzed were very simple and relied on communication to Command and Control servers to receive further directives.

Bijmans et al. [5] have performed a recent large study on organized cryptojacking. They discovered that cryptojacking campaigns have been heavily underestimated in previous studies, and that third-party software such as Wordpress is the new preferred method of spreading infections. After having crawled about 20% of the Internet, they estimate cryptomining without user consent in 0.011% of all domains, mostly prevalent in adult content sites. They also describe numerous hiding techniques present in scripts making them more difficult to detect by blocking application.

6 Conclusion

Related to our first research question, the experiment measurements show that native mining clearly outperforms web mining. Though relatively simple devices such as the Raspberry Pi had the highest hashrate per Watt, mining simply takes too much time on these that they are desirable targets. When we measured performance and latency during mining using an objective benchmarking tool, these values went down as expected as we added more mining threads, making cryptojacking easily detectable on an already stressed device. However, we got a somewhat different impression when the devices had more casual usage patterns involving video streaming, office apps, surfing and games. On devices with many available threads, the mining was hardly noticeable as long as the algorithm did not take all available resources. Since most regular users are accustomed to natural performance variations, it can therefore be very difficult to naturally recognize cryptomining running in the background.

By addressing our second research question we saw that it was difficult to justify a sound attacker business model. There was a relatively large marked for it up until 2018, but as the cryptocurrencies fell in value, the cyber criminals started to revert back to other, more profitable ways of making a revenue. It is important to remember that if the cryptocurrency markets should resurge, it is likely that cryptojacking will follow suit. The attacks are relatively easy to carry out, and since they seldom cripple the infected devices, users might not detect the mining or bother to do something about it. Luckily, the security industry is now more aware of this threat and there are many tools that can protect the users.

References

1. Al Hajri, H.H., Al Mughairi, B.M., Hossain, M.I., Karim, A.M.: Crypto jacking a technique to leverage technology to mine crypto currency. Int. J. Acad. Res. Bus. Social Sci. **9**(3), 1210–1221 (2019)
2. Alexa. https://www.alexa.com/. Accessed 23 Aug 2019
3. Analyze power use with battery historian. https://developer.android.com/topic/performance/power/battery-historian. Accessed 24 May 2019
4. App store review guidelines. https://developer.apple.com/app-store/review/guidelines/. Accessed 22 Aug 2019
5. Bijmans, H.L., Booij, T.M., Doerr, C.: Inadvertently making cyber criminals rich: a comprehensive study of cryptojacking campaigns at internet scale. In: 28th USENIX Security Symposium (USENIX Security 19), pp. 1627–1644 (2019)
6. Carlin, D., Burgess, J., O'Kane, P., Sezer, S.: You could be mine (d): the rise of cryptojacking. IEEE Secur. Priv. **9**(3), 1210–1221 (2019)
7. Coinhive.com. https://coinhive.com/. Accessed 8 Apr 2019
8. Discontinuation of coinhive (2019). https://coinhive.com/blog/en/discontinuation-of-coinhive. Accessed 24 May 2019
9. Top 100 cryptocurrencies by market capitalization (2019). https://coinmarketcap.com. Accessed 22 Aug 2019

10. Mine monero from your browser. https://coinwebmining.com/browser-miner/monero. Accessed 24 May 2019
11. Cova, M., Kruegel, C., Vigna, G.: Detection and analysis of drive-by-download attacks and malicious Javascript code. In: Proceedings of the 19th International Conference on World Wide Web, pp. 281–290. ACM (2010)
12. Cox, J.: Creators of in-browser cryptocurrency miner 'coinhive' say their reputation couldn't be much worse. https://motherboard.vice.com/en_us/article/vbpbz4/creators-of-in-browser-cryptocurrency-miner-coinhive-say-their-reputation-couldnt-be-much-worse (2018). Accessed 24 May 2019
13. Cpuminer-multi. https://github.com/tpruvot/cpuminer-multi. Accessed 24 May 2019
14. Dean, M.: 5 best cryptojacking blockers to use on your windows pc. https://windowsreport.com/cryptojacking-blockers/. Accessed 24 May 2019
15. Deep miner. https://github.com/deepwn/deepMiner. Accessed 24 May 2019
16. Eskandari, S., Leoutsarakos, A., Mursch, T., Clark, J.: A first look at browser-based cryptojacking. In: 2018 IEEE European Symposium on Security and Privacy Workshops (2018)
17. Internet Organised Crime Threat Assessment (IOCTA) 2018 (2018). https://www.europol.europa.eu/activities-services/main-reports/internet-organised-crime-threat-assessment-iocta-2018
18. Frigioiu, A.: Crypto miners: the rise of a malware empire (2018). https://blog.avira.com/crypto-miners-coinhive-malware-empire/. Accessed 24 May 2019
19. Hong, G., et al.: How you get shot in the back: a systematical study about cryptojacking in the real world. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1701–1713. ACM (2018)
20. Hron, M.: Protect yourself from cryptojacking. https://blog.avast.com/protect-yourself-from-cryptojacking (2018). Accessed 23 Aug 2019
21. istat menus. https://itunes.apple.com/us/app/istat-menus/id1319778037?mt=12. Accessed 24 May 2019
22. Kharraz, A., et al.: Outguard: detecting in-browser covert cryptocurrency mining in the wild. In: The World Wide Web Conference, pp. 840–852. WWW 2019, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3308558.3313665
23. Konoth, R.K., et al.: Minesweeper: an in-depth look into drive-by cryptocurrency mining and its defense. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1714–1730. ACM (2018)
24. Liao, S.: Calendar app in MAC app store mines cryptocurrency in the background (2018). https://www.theverge.com/2018/3/12/17110810/apple-app-store-mac-cryptocurrency-monero-calendar-2-qbix. Accessed 22 Aug 2019
25. Monero miner for web browsers. https://minero.cc/. Accessed 24 May 2019
26. Monero.com. https://monero.org/. Accessed 24 May 2019
27. Musch, M., Wressnegger, C., Johns, M., Rieck, K.: Web-based cryptojacking in the wild (2018). arXiv preprint arXiv:1808.09474
28. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf. Accessed 23 Aug 2019
29. Nguyen, V.L., Lin, P.C., Hwang, R.H.: Web attacks: defeating monetisation attempts. Netw. Secur. **2019**(5), 11–19 (2019)
30. Norman, J.: How not to become a crypto-jacking statistic. Comput. Fraud Secur. **2019**(4), 18–19 (2019)
31. Papadopoulos, P., Ilia, P., Markatos, E.P.: Truth in web mining: measuring the profitability and cost of cryptominers as a web monetization model (2018). arXiv:1806.01994v1. Accessed 24 May 2019

32. Partz, H.: Coinhive code found on 300+ websites worldwide in recent cryptojacking campaign (2018). Accessed 24 May 2019
33. Pastrana, S., Suarez-Tangil, G.: A first look at the crypto-mining malware ecosystem: a decade of unrestricted wealth. arXiv preprint (2019) arXiv:1901.00846
34. Pedersen, P.: The open source community as a top 100 country. http://www.inside-open-source.com/2007/11/open-source-community-as-top-100.html. Accessed 23 Aug 2019
35. Rüth, J., Zimmermann, T., Wolsing, K., Hohlfeld, O.: Digging into browser-based crypto mining. Chair of Communication and Distributed Systems, RWTH Aachen University (2018). Accessed 24 May 2019
36. Saad, M., Khormali, A., Mohaisen, A.: End-to-end analysis of in-browser cryptojacking (2018). arXiv:1809.02152v1. Accessed 24 May 2019
37. Saberhagen, N.V.: Cryptonote v 2.0 (2013). Accessed 24 May 2019
38. Sigler, K.: Crypto-jacking: how cyber-criminals are exploiting thecrypto-currency boom. Comput. Fraud Secur. **2018**(9), 12–14 (2018). https://doi.org/10.1016/S1361-3723(18)30086-1. http://www.sciencedirect.com/science/article/pii/S1361372318300861
39. Supportxmr.com. https://www.supportxmr.com/. Accessed 24 May 2019
40. Internet security threat report (2019). https://www.symantec.com/en/sg/security-center/threat-report. Accessed 24 May 2019
41. Sysbench. https://github.com/akopytov/sysbench. Accessed 24 May 2019
42. Tahir, R., Durrani, S., Ahmed, F., Saeed, H., Zaffar, F., Ilyas, S.: The browsers strike back: countering cryptojacking and parasitic miners on the web. In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp. 703–711. April (2019). https://doi.org/10.1109/INFOCOM.2019.8737360
43. Historical trends in the usage of client-side programming languages for websites. https://w3techs.com/technologies/history_overview/client_side_language/all. Accessed 24 May 2019
44. Plugin tag: mining. https://wordpress.org/plugins/tags/mining/. Accessed 24 May 2019
45. Wu, L.: Monero-mining hiddenminer android malware can potentially cause device failure. TrendMicro. Accessed 24 May 2019
46. Xmr-stak: Cryptonight all-in-one mining software. https://github.com/fireice-uk/xmr-stak. Accessed 24 May 2019
47. Zimba, A., Wang, Z., Chen, H., Mulenga, M.: Recent advances in cryptovirology: state-of-the-art crypto mining and crypto ransomware attacks. KSII Trans. Internet Inf. Syst. (TIIS) **13**(6), 3258–3279 (2019)
48. Zmudzinski, A.: Monero developers consider adopting new proof-of-work algorithm in october. Cointelegraph (2019). https://cointelegraph.com/news/monero-developers-consider-adopting-new-proof-of-work-algorithm-in-october. Accessed 23 Aug 2019

My Smartwatch Is Mine – Machine Learning Based Theft Detection of Smartwatches

Christian Roth(✉), Mirja Nitschke, Christian Hutzler, Maximilian Koller, Rolf Küffner, Marc Roßberger, and Doğan Kesdoğan

University of Regensburg, Regensburg, Germany
{christian.roth,mirja.nitschke,christian.hutzler,maximilian.koller, rolf.kuffner,marc.rossberger,dogan.kesdogan}@ur.de

Abstract. Smartwatches are small but powerful devices which make daily life easier and are without a doubt desirable objects for thieves. In this paper, we present a first machine learning based theft detection approach running in a user's domain, relying solely on data of his smartwatch and thus not involving third parties. Hence, we collect data from multiple persons to first show that there is an exploitable structure within data provided by a smartwatch's inertial sensors and perform user identification on the basis of that data. Then we will present and thoroughly evaluate our robust, efficient and fast (within seconds) theft detection algorithm which has both a low false rejection rate and an even lower false acceptance rate.

Keywords: User recognition · Smartwatch · Clustering · Privacy

1 Introduction

While the smartphone market declined for the sixth consecutive quarter [5], smartwatches continue to gain attention with 41.5 million smartwatches being sold worldwide in 2017. Sales are expected to almost triple in 2022 to 115.2 million units [13]. Nevertheless, the size of a smartwatch poses a particular challenge. Both the screen and the hardware must be small enough to wear the watch on your wrist. These restrictions result in the restriction of the user interface, reduced computing power, lower battery capacity and less precise sensors. These challenges also exist with smartphones, but not to the same extent. Nevertheless, smartwatches have decisive advantages over smartphones. The sensors are able to read out more data, especially due to their unique wearing position and permanent connection to the owner's skin. The wearing position plays a decisive role in activity detection. While smartphones are often transported in trouser pockets, handbags or backpacks, which can impair the sensor data read out, smartwatches are located on the wearer's wrist. This eliminates the need to first determine the wear position of the device. It is only necessary to distinguish between left- and right-handed users.

A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 171–187, 2019.
https://doi.org/10.1007/978-3-030-35055-0_11

However, the high price, increasing popularity, and size of a smartwatch also bring dangers with it. These characteristics make the device a sought-after object among thieves. Therefore, this work will investigate whether a theft of the smartwatch can be detected with the help of the integrated sensors. The sensor data read out is used to determine differences between the rightful owner and the potential thief.

Contribution. To the best of our knowledge, we are the first ones to apply theft detection to the field of smartwatches. In fact, we contribute with

– a smartwatch application for Wear OS to collect sensor data,
– a real world dataset of smartwatch sensor data from multiple persons performing activities,
– a feasibility proof by applying machine learning based user identification to our dataset which works as a baseline for our theft detection algorithm, and
– a fast and robust machine learning based algorithm for theft detection with a low footprint in terms of training data.

We also provide an exhaustive evaluation of our theft detection approach by analyzing multiple parameters such as the number of features, sensors used, and we find the best fit for balancing false alarms and usability.

Structure. First, we present our environment in Sect. 2 and talk about constraints when it comes to the usage of sensor data from smartwatches. Section 3 elaborates and shows the existence of a structure in our dataset by performing user identification. Next, Sect. 4 tackles the problem of theft detection under the given constraints. Related work (see Sect. 5) takes a brief look at the usage of sensor data, in particular inertial sensor data, in the domain of smartwatches. Last, Sect. 6 concludes our work.

2 Environmental Constraints

Smartwatches provide a multitude of sensors to tackle daily tasks. For instance, GPS can be used to track the location of a user to create a location trajectory while running with the heart rate monitor gaining information of the user's performance.

However, modern smartwaches also have inertial sensors such as gyroscope and accelerometer. These two sensors are present in almost every smartwatch. Thus, the presented approach only exploits data from these both specific sensors to detect a theft. Our system is designed to detect if a smartwatch gets stolen and carried away. This requirement is obvious since some activity has to be recorded to differentiate between multiple users.

2.1 Activity Recognition

Not all user activities are feasible for user identification and thus theft detection in real-time. We select running or walking to be the activity of choice since it is likely that a thief runs away with the stolen device. Furthermore, both are activities everyone does on a regular basis. We can therefore derive a first requirement that is needed in the chain of theft detection. First, the system has to gain information about the current activity class in order to perform more thorough inspections of the collected data. Our work is based on existing work to identify the current activity to match it against known data of individuals (e.g. [9,11]).

A system designed for theft detection of smartwatches has to be robust against activity variability. This is due to the fact that the same activity performed by different individuals can lead to different patterns. However, the system still has to identify the same activity, for instance, running. In fact, activity variability is a challenge even for the same person performing the same activity at different times. Reasons might be stress, fatigue or the emotional condition of a user. Identifying the correct activity on the basis of sensor data is not a trivial task since human moving patterns are highly complex and diverse. This task has been of interest for many years. [3] provide and evaluate a framework specifically for this task.

2.2 Imbalance of Training and Test Data

Activity recognition and eventually theft detection is no different to other machine learning tasks. The more training data is available, the better the recognition. The variability of the data has to be ensured when collecting such data to reflect real world conditions and thus being able to generate a robust model (e.g. robust against overfitting). Similar data readings can lead to the same activities, despite not being the same. For example, smoking and drinking a cup of coffee show similar data recordings [3], although both tasks are very different except that the hand moves to the mouth. The granularity of the activities can also be scaled from coarse to fine, resulting in even more diverse datasets.

In the context of theft detection, gaining huge amounts of training data is not a feasible approach, because it would require a user to perform a training phase with his smartwatch where he performs different activities in varying manners. This might have a great impact in terms of acceptance. Thus, our approach is designed to only use a very short learning period of about a few minutes, where a user just carries the smartwatch. This can be done in his daily routine.

For theft detection, just a small amount of the recorded data might provide enough evidence for a theft. Therefore, irrelevant data has to be ignored in order not to lead to false assessment of the situation. For instance, climbing the stairs is not to be confused with walking.

2.3 Environmental Constraints

In addition to the challenges of activity recognition and dataset size, other factors such as the properties of the sensors used must also be taken into account. For instance, sensors of portable devices must consider where the device is worn and how accurate measurements of these sensors are. These factors can have a significant impact on the data collected. In this context, the latency of the system is of great importance when processing the data. Many applications, such as gesture-based input or theft detection, require real-time data processing and classification to provide immediate feedback to the user. We noticed that some sensors have a high delay in terms of data provision. For instance, the heart rate sensors took more than a minute to deliver new data readings. Such sensors are conflicting with real-time requirements and thus are unfeasible for usage in this context. The battery life of the device is also limited and the approach has to be designed to work in an efficient way.

2.4 Data Collection

Both for the recognition of activities and for the identification of individuals, sensor data of physical activities must be collected in the first step. We therefore developed a Google Wear OS application which collects data for a specific user and a predefined activity. Activity recognition is not focused in this work and we want to prevent any degradation in the theft detection process because of wrongly detected activities.

We focused on data from the accelerometer and gyroscope along all three ordinal axes x, y, z. The target frequency of the collection was 50 Hz resulting in around 3000 data readings per minute. It has to be noted that with the device we used, a Huawei Watch 2, readings from the gyroscope came in slower compared to the accelerometer (only around 25 Hz). This seems to be hardware constrained.

Data was collected from 7 test persons performing different activities in order to generate a broad, diverse dataset. In total, we were able to collect around 10 h. Furthermore, one of the test persons was left-handed. We explicitly selected such a person to see how our theft detection approach can handle left-handed and right-handed people and to further analyze if there are significant differences between both groups.

2.5 Data Preprocessing

The sensors deliver a continuous data stream, whereby each measured value represents the state of a sensor at a certain point in time. In the data segmentation step, individual segments are extracted from the data stream that are highly likely to contain specific information about the motion of an individual's activity. This step is primarily performed to apply statistical techniques to specific parts of the data stream. A widely used approach to data segmentation is the so-called sliding window. In this approach, a window of a fixed size is moved over the data in a certain increment. The step size is selected so that the individual windows overlap each other. The selection of the window size is particularly important for real-time applications, as a window that is too large greatly impairs the latency of the process.

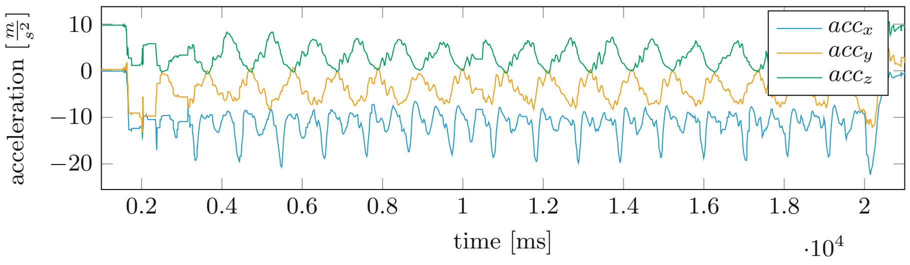


Fig. 1. Excerpt of data readings of a person walking in a straight line showing a clear, reoccurring pattern.

Next, we extract features for each window eventually leading to individuals. A common approach is to generate these features based on statistical measures such as minimum, maximum and standard deviation. Our experiments show that using more features does not necessarily increase the detection performance. Because of the limited environment, we tried to minimize the number of features used during the theft detection process. In fact, the following features were extracted:

1. Minimum 2. Maximum 3. Range 4. Average 5. Median 6. Sum of squares 7. Variance 8. Standard deviation 9. Median absolute deviation.

The use of extracted features in time windows also has the advantage that the amount of data sets used for training a model can be greatly reduced. For example, when using raw sensor data with a frequency of 50 Hz, 3000 data records per minute are generated. For window lengths ranging from 2 s to 10 s, the number of elements used for processing and identification can be reduced to 12–60 data sets. This leads to an acceleration of the training phase, which is particularly relevant on a smartwatch with reduced computing power.

3 Distinction of Individuals

In order to find any structure within the data, we performed user identification on the basis of the sensor data from the smartwatch. We used data from up to 7 participants to validate our findings.

3.1 Visual Analysis

Acceptable theft detection requires that one individual is at least identifiable with high accuracy using sensor data to ensure low false positive and false negative prediction rates in the theft scenario.

Figure 1 shows the accelerometer sensor reading for a person walking in a straight line. At the beginning and the end, only gravity (around $9.8\frac{m}{s^2}$) affects the accelerometer, though when moving, arm movement introduces a second impact. It is possible to identify a periodic pattern resulting from a swinging arm while walking. This underlines the fact that one person seems to have a reoccurring pattern while performing an activity. However, the amplitude is not always identical between swings. As a consequence, anomaly detection for theft detection will not be a feasible approach.

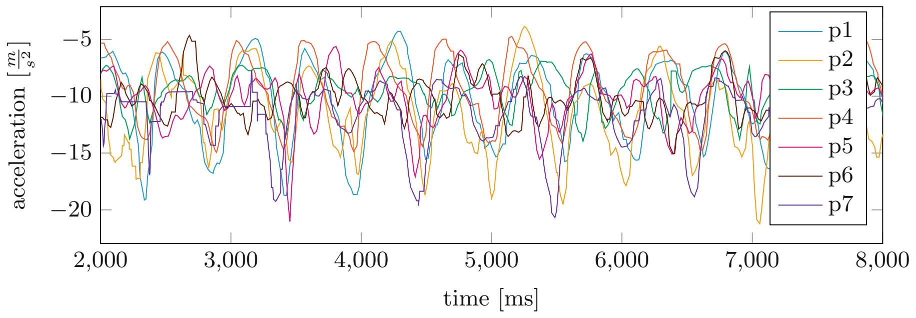


Fig. 2. Excerpt of sensors readings of all persons in the data set walking straight (x-axis is shown).

To further analyze if there is even a difference between the walking style and the resulting data reading of multiple people, we plotted randomly selected walks from all participants in Fig. 2. One can easily see that there are a lot of differences between all participants, such as the amplitude or the length of a curve. Reasons for that are the walking speed or the arm swinging style.

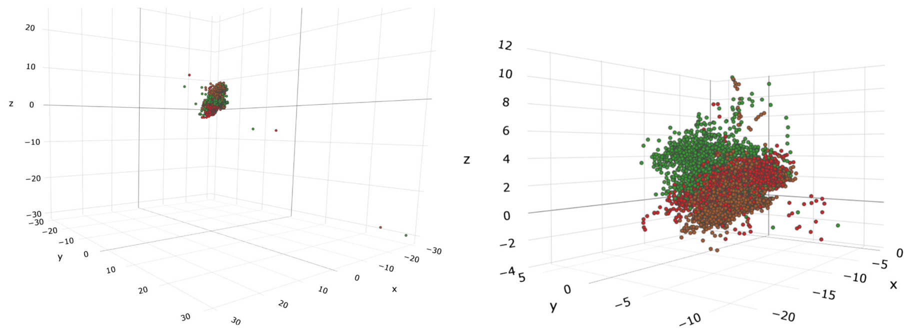
Table 1. Performance overview of four different classification algorithms.

Algorithm	Accuracy [%]
Naive Bayes	75.5
Decision Tree J48	96.6
k-Nearest-Neighbor	97.4
Random Forest	98.4

We used the raw data set to confirm that the recorded data allows user identification by having a (hidden) structure. We therefore performed identification using only a one minute testing sample of each user. Further optimization of the data was not done. Table 1 shows promising results that identification is possible in our dataset.

3.2 Cluster Analysis on Raw Data

In a first attempt, the collected data was clustered in order to find similarities across three random test subjects (each represented by a color in Figs. 3a, b and 4a to d).



(a) Clustered raw gyroscope data. (b) Clustered raw accelerometer data.

Fig. 3. Three clusters of raw data for three test subjects across all axes (each using own color). (Color figure online)

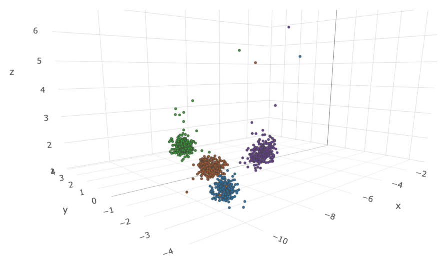
Figure 3a points out that all three gyroscope-based clusters are diverse, i.e. they are mixed up and no sharp borders can be found which can be exploited to determine a users. This is very different for readings from the accelerometer as illustrated in Fig. 3b. Especially the z-axis can be considered as an indicator. The green colored datasets are rather in the upper area, the brown colored ones in the lower area. However, the centers of the different clusters are very close to each other, which, according to the objectives of the cluster analysis, could make it difficult to correctly classify a dataset between them.

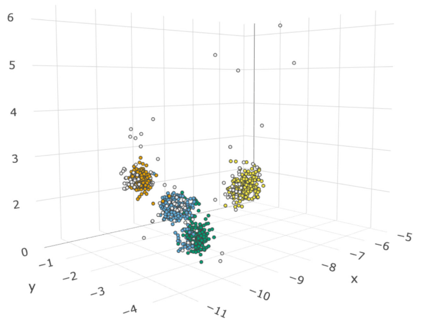
We then added a noise cluster to collect all data not exactly assignable to a user cluster. As a result, most of the test data from both sensors was assigned to that cluster, indicating that there is no clear structure within the raw data. Most data seems to be randomly adjusted to a cluster making the usage of raw data for theft detection not feasible. Parameter tuning did not increase the quality.

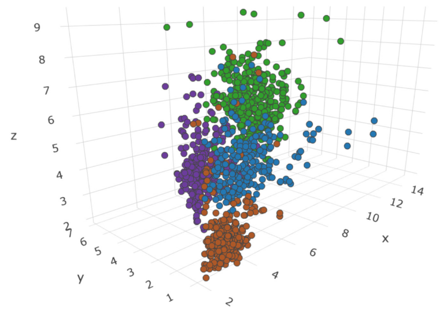
3.3 Cluster Analysis with Preprocessing

Thus, we decided to use the state of the art approach and subdivided the continuous data stream in windows using the sliding window approach. We selected a window size of 5 s with an overlapping of 50%. We further extracted the features stated in Sect. 2.5. All experiments were executed using four random test participants.

Figure 4a and c present the average acceleration across all axes respectively the range of the gyroscope readings for each window of each individual. In contrast to the raw readings, the preprocessed dataset shows clear cluster centers and thus allows to divide and identify the users. It has to be noted that although the figures only show one feature, we used all said features to identify a users. Introducing a noise cluster and taking a look at the results (c.f. Fig. 4b and d; noise is represented as white dots), it can be stated that the noise cluster has an inferior role and there is a structure within the data which may be exploited for identification.


(a) Clustered average acceleration across all windows.


(b) Clustered average acceleration across all windows with a noise cluster.


(c) Clustered range of gyroscope across all windows.

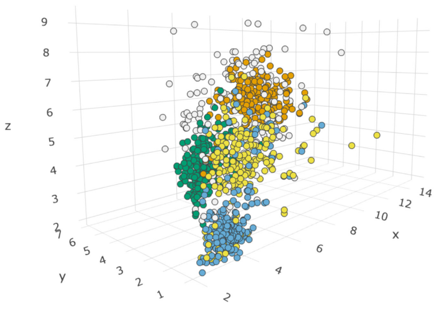

(d) Clustered range of gyroscope across all windows with a noise cluster.

Fig. 4. Three clusters of preprocessed data divided in windows and all features for three test subjects across all axes (each using own color). (Color figure online)

Now, we aggregate our findings by performing the actual identification process. Even if there are clusters within the dataset, it cannot be assured that all data points are assigned to the correct cluster (i.e. successful identification).

Table 2 presents the confusion matrix for our identification model. The goal is to assign a user to one cluster which represents that user. It can be seen that identification is possible with high accuracy of 88% overall. This shows that the sensor data from smartwatches with the correct preprocessing allows correct identification and is therefore a good basis for theft detection. However, there are some false predictions which have to be handled in the context of theft detection.

It should also be mentioned that different window sizes were tested and led to similar results. Furthermore, there are different methods to determine the influence of the individual attributes per dataset on the classification in a specific cluster. We used a visual approach to identify significant attributes such as the mean acceleration along the x-axis and the average absolute distance from mean along the x-axis. The cluster centers of these two attributes varied significantly between the four participants.

4 Theft Detection

After having analyzed if user identification on the basis of smartwatch data is even possible, we now want to aggravate the problem and perform theft detection almost in real time. This problem is different from user identification since we move from a closed set problem to an open set problem where not all carriers of the watch are already known.

Table 2. Confusion matrix of user identification.

		Predicted			
		cluster_0	cluster_1	cluster_2	cluster_3
Actual	user_0	0.08	0.81	0.11	0.00
	user_1	0.00	0.00	0.00	1.00
	user_2	0.00	0.18	0.79	0.04
	user_3	0.91	0.09	0.00	0.00

4.1 Definition of Environment

We construct two different scenarios targeting different research questions.

Scenario 1. The first scenario examines whether it *is possible to distinguish a person from other test subjects if only a small amount of training data is available*. For this scenario, 2 data records of a selected owner of one afternoon of about 10 min each are used as training data. As test data, data records of different persons are used as thieves, with a total length of 2 h on the same and other test tracks, as well as a data record of the owner from the morning of the training day with a length of 7 min on the same track were used.

Scenario 2. The purpose of the second scenario is to simulate the use of a theft detection app and *to check how much data is needed to correctly classify an owner and thieves*. For this purpose, training data was collected over several days from one owner on different tracks and under different weather conditions with a total length of 67 min. Due to significant differences in walking behavior depending on the day, weather and possibly other factors, it is difficult to recognize an owner on a new day. Thus, the owner's training data is randomly divided into test and training data and used for evaluation. Additionally, movement data over a length of 82 min of other persons used as thieves was collected to test the model.

Both scenarios have in common that we do not focus on different activities but use "walking" as a common activity. However, this constraint can easily be widened by applying an additional filter at preprocessing to select the appropriate training data/model for the recognized activity.

For an evaluation of the classification model, we use a binary confusion matrix from which two error rates can be read. First, if the owner wears the watch and is falsely classified as a thief, it is a false rejection, the rate of this error is specified in the *False Rejection Rate* (FRR) and represents a false alarm in the application scenario. The second error is that a thief is mistakenly classified as the owner and it is a false acceptance, represented by the *False Acceptance Rate* (FAR), which is the more serious error as it would cause a theft to go unnoticed.

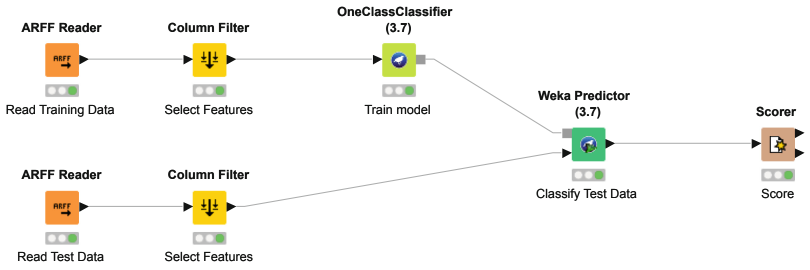


Fig. 5. KNIME workflow for Scenario 1.

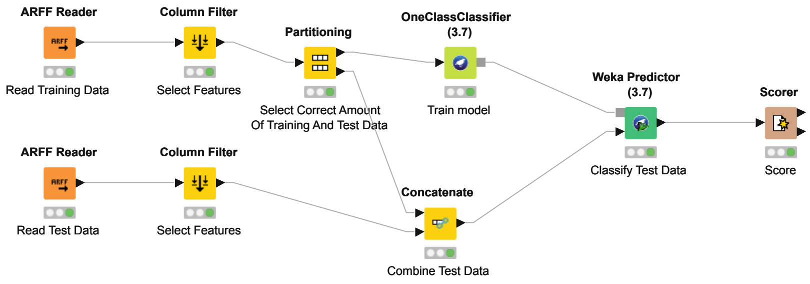


Fig. 6. KNIME workflow for Scenario 2.

4.2 Theft Detection Accuracy

Classification was performed using a One-Class-Classifier combining density and class probability estimation [7] (KNIME module version 3.7) provided by the KNIME Analytics Platform (version 4.0.1). All features were used, which results in 63 features per window. We also tested different window sizes, however overlapping was fixed to 50%. One individual from our dataset was selected to be the smartwatch's rightful wearer, while all other participants from the dataset were considered to be thieves. We repeated the experiments.

Figure 5 shows the KNIME workflow for Scenario 1 which used a fixed number of training samples. In contrast, Fig. 6 illustrates the workflow for the second scenario which differs in the amount of data used for training. The remaining data of the owner was used for testing the model.

From Table 3 it is easily recognizable that the approach which uses raw data is unfeasible for theft detection because, in both scenarios, thieves are wrongly classified as owners (high FAR). As a result, we do not further consider such an approach. In Scenario 2, thieves as well as owners can be identified correctly using extracted features. A positive observation is that already on the basis of 10% training data the remaining 90% of the owner's movement data are correctly classified in approximately 90% of the cases. If preprocessing is performed, the FAR is always very low, thus a thief is not confused with the owner.

Table 3. Comparison of the suitability of raw data and extracted features for classification purposes.

	R[a]		F,$ws = 2s$[b]		F,$ws = 5s$[b]		F,$ws = 10$[b]	
	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
Scenario 1	65.8	5.8	1.0	18.8	0.2	41.1	0.1	90.4
Scenario 2								
.. 10% training	48.5	10.1	0.8	7.7	0.2	8.0	4.1	7.0
.. 25% training	48.4	10.3	0.4	8.8	0.1	10.5	0.0	15.8
.. 50% training	48.7	10.0	0.2	9.8	0.2	8.9	0.0	15.5
.. 75% training	48.6	10.2	0.4	7.8	0.1	11.4	0.7	9.6

All numbers given in percent

[a]Raw data as it comes from the smartwatch sensors

[b]Preprocessed data with all features and window size ws

For this evaluation, the standard configuration of the One-Class-Classifier was applied, with the exception that only a kernel density estimation was used for the training of the model, since the training of a model in another configuration requires more computing capacity. For example, when using all 208,000 owner data records of Scenario 2, training the model took 30 min compared to only a few seconds when applying only kernel density estimation.

4.3 Adjustment of Target Rejection Rate

Using features for classification in Scenario 1 results in a high false rejection rate. This is due to the differences in the owner's walking movements between training data and test data. To test whether classification is possible despite such differences, the *Target Rejection Rate* (TRR) of the classifier was reduced from 10% to 5% in a further test. The classifier uses this rate to determine what portion of the owner's (target's) existing training data should be classified as non-owner to form a boundary between target and non-target data. Using this boundary, the model can evaluate new data and decide whether the data set is within the owner's boundary or should be classified as a thief. If the test data is identical to the training data, a portion of the data that corresponds to the

TRR is also classified as a thief. As a result, a perfect model with a FAR of 0% will have a FRR corresponding to the TRR as expected. Since in Scenario 1 a large part of the owner's data is classified as a thief, it was examined whether an adaptation of the TRR would allow an improvement in owner recognition. In addition, Scenario 2 was used to check whether such a change can also be used in a model that was trained with more diverse data sets.

Table 4. Results of the classification when using a TRR of 5%.

	F,$ws = 2s$		F,$ws = 5s$		F,$ws = 10$	
	FAR	FRR	FAR	FRR	FAR	FRR
Scenario 1	6.5	4.8	1.5	7.8	0.5	17.4
	(↑5.5)[a]	(↓14.0)	(↑1.3)	(↓33.3)	(↑0.4)	(↓73.0)
Scenario 2						
.. 25% training	12.3	3.5	9.5	3.0	0.0	10.8
	(↑11.5)	(↓4.2)	(↑9.3)	(↓5.0)	(↓4.1)	(↑3.8)
.. 75% training	10.3	3.4	5.9	5.3	28.6	6.1
	(↑9.9)	(↓4.4)	(↑5.8)	(↓6.1)	(↑27.9)	(↓3.5)

All numbers given in percent

[a]Absolute change in comparison to Table 3

The results (c.f. Table 4) illustrate the influence of the TRR. By lowering the number of records classified as owners, the FRR decreases in all cases, but the FAR increases. For Scenario 1, this adjustment is an improvement since both error rates are now within an acceptable range in the case of the 2 and 5 s window, but when transferring the setting to Scenario 2, the FAR is greatly increased without generating much added value by reducing the FRR. For these reasons, a reduction in TRR seems inappropriate when used in reality and a TRR of 10% will be used subsequently.

4.4 In-Depth Analysis of Our Approach

We now want to analyze different input vectors of the theft detection approach to understand the prediction accuracy. Namely we analyze if the number of features is relevant for the detection and which sensor provides more information in the context of theft detection. Lastly, we loosen our constraint to only rely on kernel density estimation and use several state-of-the-art classifier models. Only Scenario 2 will be considered here because it describes the theft detection.

Features. First, we change the number and types of features used for theft detection. Table 5 shows some interesting facts. The existing model seems to use too many features because a similar theft detection accuracy can also be achieved only using averages, minima and maxima from the sensors. This sensor selection even comes to a lower FAR without negatively influencing the FRR. This suggests that using too many features leads to some kind of overfitting.

Sensors. Literature has shown that accelerometer data is sufficient to detect an activity [9], but identification requires a second sensor such as the gyroscope [8]. Thus, we want to elaborate if theft detection is also possible using only accelerometer or gyroscope. In the aftermath of the previous section, we only execute our experiments using average-, minima- and maxima-based features. From Table 6, one can see that the gyroscope improves the already good results from the accelerometer. So we can conclude that using both sensors is recommended.

Table 5. Results of different feature sets.

	F,$ws = 2s$		F,$ws = 5s$		F,$ws = 10$	
	FAR	FRR	FAR	FRR	FAR	FRR
25% training						
All features	0.4	8.8	0.1	10.5	0.0	15.8
Only averages	3.4	8.5	0.7	11.0	0.7	14.7
	(↑3.0)[a]	(↓0.3)	(↑0.6)	(↑0.5)	(↑0.7)	(↓1.1)
Average, min/max	0.0	9.1	0.0	12.7	0.0	14.6
	(↓0.4)[a]	(↑0.3)	(↓0.1)	(↑2.2)	(→0.0)	(↓1.2)
50% training						
All features	0.2	9.8	0.2	8.9	0.0	15.5
Only averages	4.2	7.5	5.9	8.9	0.1	17.8
	(↑4.0)	(↓2.3)	(↑5.7)	(→0.00)	(↑0.1)	(↑2.3)
Average, min/max	0.0	8.7	0.1	8.1	0.0	16.0
	(↓0.2)	(↓1.1)	(↓0.1)	(↓0.8)	(→0.0)	(↑0.5)
75% training						
All features	0.4	7.8	0.1	11.4	0.7	9.6
Only averages	4.7	7.0	3.7	11.1	9.0	9.1
	(↑4.3)	(↓0.8)	(↑3.6)	(↓0.3)	(↑8.3)	(↓0.5)
Average, min/max	0.1	7.9	0.1	9.9	0.2	11.2
	(↓0.3)	(↑0.1)	(→0.0)	(↓1.5)	(↓0.5)	(↑1.6)

All numbers given in percent

[a]In comparison to all features

Classifiers. All previous results are achieved using kernel density estimation which is a robust approach for One-Class classification. However, we wonder if a hybrid approach (which eventually increases training time, though) using another classifier will improve the already good performance of our theft detection algorithm. Table 7 gives an overview of the performance of three classifiers, namely Naive Bayes, BayesNet and a Logistc Regression model. The other classifiers provide similar performance, although their training time is much longer compared to kernel density estimation.

4.5 Discussion

Section 4 shows that a One-Class-Classification is possible on the basis of collected data of the owner of a smartwatch. It was shown that an identification is already possible with a small dataset (Scenario 1), but can be significantly improved by collecting further data (Scenario 2). The walking behavior of a person may change, but even a small amount of data (around 25%) of one movement pattern is sufficient to successfully train the model. We were able to achieve a FAR $\leq$ 0.2% while keeping an acceptable FRR below 8% with the FRR only limited by the TRR (which could still be optimized).

Table 6. Results of different sensor combinations.

	F,$ws = 2s$		F,$ws = 5s$		F,$ws = 10$	
	FAR	FRR	FAR	FRR	FAR	FRR
25% training						
acc + gyro	0.0	9.1	0.0	12.7	0.0	14.6
accelerometer	1.0	9.2	0.5	12.8	0.0	18.8
	(↑1.0)[a]	(↑0.1)	(↑0.5)	(↑0.1)	(→0.0)	(↑4.2)
gyroscope	13.8	9.3	7.3	10.1	1.7	17.5
	(↑13.8)[a]	(↑0.2)	(↑7.3)	(↓2.6)	(↑1.7)	(↑2.9)
75% training						
acc + gyro	0.1	7.9	0.1	9.9	0.2	11.2
accelerometer	1.0	9.2	2.0	9.4	7.3	9.6
	(↑0.9)	(↑1.3)	(↑1.9)	(↓0.5)	(↑7.1)	(↓1.6)
gyroscope	12.5	7.2	7.4	10.1	11.6	10.7
	(↑12.4)	(↓0.7)	(↑7.3)	(↑0.2)	(↑11.4)	(↓0.5)

All numbers given in percent
[a]In comparison to acc + gyro

For real world use, we derive the following rule candidate to detect a theft: *If 3 or more windows are classified as theft within the first 5 time windows, an alarm is sent.* Assuming an FRR of 10%, the probability of a false alarm decreases to 0.856%. With the same decision rule and an expected FAR of 0.5%, the risk of undetected theft is reduced to approximately 0.0001%. Since no relevant difference could be detected when evaluating the classification on the basis of time windows of different lengths, the shortest window with a length of 2 s is best suited for a time-critical application such as theft detection. Thus, by overlapping the time windows, after a time of 6 s, a good decision can already be made as to whether it is a thief or the owner of the smartwatch.

5 Related Work

The topic of activity recognition using smartwatches has been investigated in different works in recent years. [2–4,6,10–12,14] address the recognition of diverse activities with different level of complexity with various machine learning approaches. For example [4] show how the smoking behavior can be detected and how it can even be distinguished from other similar gestures. [2] evaluate the heartbeat rate and wrist acceleration data, gathered via a smartwatch, in order to identify subject's sleep behavioral pattern. In addition, research on activity recognition has already been extended to the identification of people in some studies. Individuals could be recognized on the basis of certain movements. [1,8] show that people can be identified by gait with a certain accuracy. However nobody discussed the topic of theft detection.

Table 7. Results of different classifiers.

	F,$ws = 2s$		F,$ws = 5s$		F,$ws = 10$	
	FAR	FRR	FAR	FRR	FAR	FRR
25% training						
KDE	0.0	9.1	0.0	12.7	0.0	14.6
Naive Bayes	2.7	9.1	0.0	12.4	1.2	14.1
	(↑2.7)[a]	(→0.0)	(→0.0)	(↓0.3)	(↑1.2)	(↓0.5)
BayesNet	0.0	9.0	0.0	12.7	0.0	17.1
	(→0.0)[a]	(↓0.1)	(→0.0)	(→0.0)	(→0.0)	(↑2.5)
Logistic Regression	0.0	9.3	0.0	12.0	0.0	14.4
	(→0.0)[a]	(↑0.2)	(→0.0)	(↓0.7)	(→0.0)	(↓0.2)
75% training						
KDE	0.1	7.9	0.1	9.9	0.2	11.2
Naive Bayes	0.5	9.1	0.5	8.0	0.7	11.2
	(↑0.4)[a]	(↑1.2)	(↑0.4)	(↓1.9)	(↑0.5)	(→0.0)
BayesNet	0.0	7.3	0.0	10.1	0.5	10.2
	(↓0.1)[a]	(↓0.6)	(↓0.1)	(↑0.2)	(↑0.3)	(↓1.0)
Logistic Regression	0.1	7.9	0.1	9. 1	0.2	11.2
	(→0.0)[a]	(→0.0)	(→0.0)	(↓0.8)	(→0.0)	(→0.0)

All numbers given in percent
[a]In comparison to kernel density estimation (KDE)

6 Conclusion

In this work we have provided a first and successful attempt to provide theft detection for coveted smartwatches. Thus, we are able to detect if a thief (i.e. unauthorized wearer) carries away a smartwatch and uses it. Not only does the classification of movement data of a user in closed-set environments provide good results, but rather the realistic open-set problem was also handled with high accuracy and fast detection performance making our approach feasible for daily usage. Furthermore, we constrained ourselves by limiting the amount of training data available to significantly increase user acceptance. We have shown in two realistic scenarios that real-time theft protection is possible and that our approach recognizes a theft within 6 s – a time span where a potential thief is still in sight of the owner – while minimizing false alarms.

Our algorithm may be improved using a more complex theft detection process since we did not focus on activity recognition which is a necessary fundamental for strong user identification. Human Activity Recognition (HAR) approaches already exist in literature (e.g. [9,11]). Thus we plan to include such an approach in our process in order to prefilter sensor data to improve our approach. As we have shown, a very limited amount of training data is sufficient for good results, thus, usability is still ensured. Furthermore, we want to integrate the alarming system into a real Wear OS application since there is a huge need for such an app. By integrating the whole theft detection process into a user's smartwatch (or domain) without the need for any external services, we will provide a privacy friendly solution. We further want to make our theft detection approach even more secure by analyzing how training data from different days and various moods affects the detection.

References

1. Ahmad, M., et al.: Smartwatch-based legitimate user identification for cloud-based secure services. Mob. Inf. Syst. **2018**, 14 (2018)
2. Alfeo, A.L., Barsocchi, P., Cimino, M.G.C.A., La Rosa, D., Palumbo, F., Vaglini, G.: Sleep behavior assessment via smartwatch and stigmergic receptive fields. Pers. Ubiquitous Comput. **22**(2), 227–243 (2018)
3. Bulling, A., Blanke, U., Schiele, B.: A tutorial on human activity recognition using body-worn inertial sensors. ACM Comput. Surv. (CSUR) **46**(3), 33 (2014)
4. Cole, C.A., Anshari, D., Lambert, V., Thrasher, J.F., Valafar, H.: Detecting smoking events using accelerometer data collected via smartwatch technology: validation study. JMIR mHealth uHealth **5**(12), e189 (2017)
5. Counterpoint, T.: Global Smartphone Market Share: By Quarter, June 2019. https://www.counterpointresearch.com/global-smartphone-share/
6. Garcia-Ceja, E., Brena, R.F., Carrasco-Jimenez, J.C., Garrido, L.: Long-term activity recognition from wristwatch accelerometer data. Sensors (Switzerland) **14**(12), 22500–22524 (2014)
7. Hempstalk, K., Frank, E., Witten, I.H.: One-class classification by combining density and class probability estimation. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 505–519. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87479-9_51
8. Johnston, A.H., Weiss, G.M.: Smartwatch-based biometric gait recognition. In: 2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS), pp. 1–6. IEEE, September 2015
9. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. ACM SIGKDD Explor. Newsl. **12**(2), 74 (2011)
10. Nguyen, M., Fan, L., Shahabi, C.: Activity recognition using wrist-worn sensors for human performance evaluation. In: Proceedings of the 15th IEEE International Conference on Data Mining Workshop, ICDMW 2015, pp. 164–169 (2016)
11. Ravi, N., Dandekar, N., Mysore, P., Littman, M.L.: Activity recognition from accelerometer data. In: AAAI, vol. 5, pp. 1541–1546 (2005)

12. Shoaib, M., Scholten, H., Havinga, P.J., Incel, O.D.: A hierarchical lazy smoking detection algorithm using smartwatch sensors. In: 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services, Healthcom 2016 (2016)
13. statista: Prognose zum Absatz von Smartwatches weltweit in den Jahren 2016 bis 2022 (in Millionen Stück) (2018). https://de.statista.com/statistik/daten/studie/500483/umfrage/prognose-zum-weltweiten-absatz-von-smartwatches/
14. Sun, X., Qiu, L., Wu, Y., Cao, G.: ActDetector: detecting daily activities using smartwatches. In: Proceedings of the 14th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2017, pp. 1–9 (2017)

Even Turing Should Sometimes Not Be Able to Tell: Mimicking Humanoid Usage Behavior for Exploratory Studies of Online Services

Stephan Wiefling[1](✉), Nils Gruschka[2], and Luigi Lo Iacono[1]

[1] TH Köln - University of Applied Sciences, Cologne, Germany
{stephan.wiefling,luigi.lo_iacono}@th-koeln.de
[2] University of Oslo, Oslo, Norway
nilsgrus@ifi.uio.no

Abstract. Online services such as social networks, online shops, and search engines deliver different content to users depending on their location, browsing history, or client device. Since these services have a major influence on opinion forming, understanding their behavior from a social science perspective is of greatest importance. In addition, technical aspects of services such as security or privacy are becoming more and more relevant for users, providers, and researchers. Due to the lack of essential data sets, automatic black box testing of online services is currently the only way for researchers to investigate these services in a methodical and reproducible manner. However, automatic black box testing of online services is difficult since many of them try to detect and block automated requests to prevent bots from accessing them.

In this paper, we introduce a testing tool that allows researchers to create and automatically run experiments for exploratory studies of online services. The testing tool performs programmed user interactions in such a manner that it can hardly be distinguished from a human user. To evaluate our tool, we conducted—among other things—a large-scale research study on Risk-based Authentication (RBA), which required human-like behavior from the client. We were able to circumvent the bot detection of the investigated online services with the experiments. As this demonstrates the potential of the presented testing tool, it remains to the responsibility of its users to balance the conflicting interests between researchers and service providers as well as to check whether their research programs remain undetected.

Keywords: Black box testing · Evaluation · Testing framework

1 Introduction

The advancing digital transformation impacts all areas of human life. As a consequence, research aiming at understanding the inner workings of digital technologies, platforms, applications, services, and products, as well as their influence on human society and culture, becomes increasingly important.

A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 188–203, 2019.
https://doi.org/10.1007/978-3-030-35055-0_12

Numerous examples for such intersections of online services with society can be found today. All sorts of social networks use non-transparent algorithms to perform content filtering and provisioning tasks, depending on one's individual characteristics and interests. Other examples can be found in the various deployed recommendation systems of e-commerce and content distribution platforms. To what extent these types of online services influence society is an important research question. Is your taste in music governed by music streaming companies and their algorithms to promote or recommend music? Are these systems exploitable for purposes other than the intended ones? First research attempts indicate that such influences on society are taking place [6,32,43].

Besides the impact on culture and society, technical aspects are more and more hidden behind the user interfaces of online services. Deployed security, as well as privacy preserving and undermining technologies, remain opaque to the user. For instance, contemporary security approaches to strengthen password-based authentication with Risk-based Authentication (RBA) [18] are deployed by only a few large online services [2,23,28], even though this technology is of broad relevance as the recent recommendation by NIST emphasizes [22]. Studying RBA-instrumented services would help to demystify RBA setups so that they can be discussed and further developed by a wider audience. This may contribute to accelerate the adoption and deployment of RBA in the wild. Good examples for exploratory research that are beneficial for society are the various studies on misusing cookies for tracking purposes [7,9,12,17,26].

An essential prerequisite to perform effective and reliable research, in this context, is the availability of data. Although many openly accessible data sets of various online services and platforms exist [4], they only provide a very limited and fragmented view. One major reason for this lack of data is that the companies and organizations possessing it—most commonly—do not share it (publicly). Thus, the digital utilities surrounding our daily lives are black boxes that do not reveal their internal workings. As this lack of transparency hinders scientific research, methods are required to methodically reverse-engineer these black boxes. This is important to understand the algorithms influencing our current and future zeitgeist as well as their corresponding security and privacy features.

Unfortunately, the investigation of the inner workings of online services is complicated for several reasons which turns studying them into a difficult problem. There is no unique path to conduct such an analysis, no simple agreed Application Programming Interface (API) or even approach. Moreover, online services are distributed systems, making the service-side inaccessible to entities other than the respective service provider itself. Also, investigating the inner workings of online services is further complicated by means of the service provider. For large-scale methodological studies, automated browsing through online services is required. However, online services integrate technical countermeasures against such automated browsing. These range from presenting CAPTCHA challenges [11] or delivering different website contents for human users and bots [1], to completely blocking the service access [30]. Hence, in order to be able to conduct exploratory studies of online services, technologies are required to camouflage automated black box testing as far as possible.

This arms race between service providers and researchers lies in their contradicting requirements. Service providers want to keep their internals secret, as they might also contain intellectual property. Researchers instead, are keen to analyze and understand systems thoroughly, with the aim to gain knowledge and enhance system properties towards an optimum. Thus, in the absence of other means, researchers will use black box tests to determine their research results, while service providers will detect and block automated black box tests to keep their internals opaque to outsiders.

Another reason for service providers to block automated black box testing is that it is considered a double-edged sword since it might not only be used by researchers. In the hands of attackers, such testing tools can be used to threaten systems and networks, even if they are aimed at improving security. Still, as security is about balancing several trade-offs, these trade-offs need to be understood thoroughly in order to make the right compromise.

Contributions. We introduce an inspection tool to perform automated black box testing of online services and, at the same time, mimic human-like user behavior[1]. The aim is to provide a research vehicle to investigate the inner workings of online services lacking publicly accessible resources. This can foster discussion and collaboration among security researchers and service providers.

Outline. The rest of this paper is structured as follows. We review related work in Sect. 2. We describe the introduced inspection tool in Sect. 3. We give more detailed descriptions on its implementation as well as customization to study online services in Sect. 4. To further illustrate the use of the introduced inspection tool, Sect. 5 discusses exemplary studies. We discuss the benefits as well as limitations of the introduced inspection tool in Sect. 6. As the usage of our tool can easily be extended to exploit online services, Sect. 7 discusses ethical considerations before the paper concludes in Sect. 8.

2 Related Work

A number of researchers performed black box testing of online services with web browser automation. Choudhary et al. [10] developed a tool for automated web application testing to detect cross-browser inconsistencies on websites. Starov and Nikiforakis [34] analyzed the effect of browser extensions on the rendered Document Object Model (DOM) document for the 50 most popular websites. They showed that differences inside the DOM tree can be (mis-)used for fingerprinting and tracking the client. Englehardt and Narayanan [15] measured and analyzed one million websites and their corresponding usage of online tracking as well as the effect of browser privacy tools. Golla and Dürmuth [19] used

[1] Provided as open source software at https://github.com/das-th-koeln/HOSIT.

browser automation to test password strength meters of online services. Degeling et al. [12] automatically extracted cookie consent notices and privacy policies of 6,579 websites inside the European Union (EU) to analyze their appearance before and after the EU General Data Protection Regulation (GDPR) [16] went into effect.

However, in all publications mentioned above, the corresponding browser automation frameworks did not aim to imitate human-like behavior as we did in our framework. As a consequence, these studies cannot tell whether their observations reflect the services' inner workings or a customized behavior due to being detected as a bot.

Other browser automation frameworks tried to imitate human-like user-actions to a small extent. Petsas et al. [29] used browser automation to evaluate the quantity of Google users with enabled Two-factor Authentication (2FA). Their framework introduced a random waiting time between clicks. Snickars and Mähler [32] analyzed the behavior of the online music streaming service Spotify with browser automation. Their automation framework conducted several user actions, e.g., logging in, selecting a track, and skipping a track. However, they noted that this was only possible before Spotify introduced reCAPTCHAs as a bot protection mechanism in 2016.

In contrast to all these frameworks, we included a considerably higher amount of efforts in our framework to closely mimic human-like behavior and bypass CAPTCHAs to not be detected as a bot (see Sect. 6).

The DASH tool [13] by the DETER project aimed to model human behavior in various situations, e.g., responding to phishing emails. In contrast to our tool (see Sect. 3), the application did not really conduct human-like actions on online services and only simulated possible behavior in theory.

Most browser automation tools described in this section were based on the Selenium framework [10,12,15,17,19,32,34]. One tool was based on CasparJS [29]. We decided to use the high-level application programming library Puppeteer [21] as a base for our tool. We chose Puppeteer over Selenium since it offers a higher-level API and is targeted to the popular Chrome browser [44] instead of multiple browsers. Note that Puppeteer was not available at the time where most of the above mentioned studies were conducted[2].

3 Humanoid Online Services Inspection Tool

The Humanoid Online Services Inspection Tool (HOSIT) was designed to simulate human-like browsing behavior on online services. While some frameworks for automated browsing are freely available on the Internet, their standard functionality makes them difficult to use for inspecting online services for several reasons:

[2] First version of the source code was published on the Puppeteer GitHub repository on May 11th, 2017: https://github.com/GoogleChrome/puppeteer/commit/2cda8c18d10865d79d3e63b23e36aa7562098bf7.

- There is no function to create virtual identities which are perceived as real humans by online services.
- Some of the integrated functions do not model real-world human behavior and thus can be detected by online services, e.g., typing with 0 ms delay or clicking in the exact center of an element.
- The API allows activities which are not possible for real web browser users, e.g., conducting browsing activities inside two browser tabs at the same time.
- Browser automation using these frameworks can be detected due to differences between the normal and the automated browser mode [40].
- These frameworks do not log conducted actions such as name and screenshot of clicked elements automatically. This makes potential implementation errors (e.g., element with certain ID not found) hard to detect. Consequently, scaling the automation to multiple machines is difficult.

We addressed these issues with HOSIT and enhanced the integrated standard functionalities of Puppeteer with human-like browsing behavior and camouflage measures to be as indistinguishable from human users as possible:

(i) A scrolling function to imitate reading of website contents (usage can be seen in the script in Fig. 2). The function scrolls down around half the display height, pauses for some time, scrolls further, pauses again and repeats this procedure until reaching the bottom of the page. We developed this function since scrolling is considered a typical behavior for human users on websites [36,42].
(ii) A function which allows switching between browser tabs. This is also a typical behavior for a human using a web browser.
(iii) A search query generator based on current events in media. The generated queries can be used to create arbitrary browsing behavior, e.g., entering query in a search engine and opening one of the results. The generator accesses a publicly available Really Simple Syndication (RSS) feed and parses the feed's content to an evaluation function which generates a list of search queries. From this list, the generator selects a random entry every time the generator is called. The search query generators can be customized and added in the HOSIT configuration, e.g., for generating search queries focused on other topics. We chose this functionality since visiting search engines is a common online activity [24,27].
(iv) Hidden element checks: some online services integrate hidden elements which can be used to detect bots, e.g., typing text inside a hidden field or clicking a hidden link. For this reason, we provide a function to check whether a certain element on a website is visible or not.
(v) Integration of external services providing CAPTCHA solving capabilities.
(vi) Automated logging of all activities conducted on the online service with screenshots into a MongoDB database for replicable studies. The database type can be adjusted for individual use case scenarios.

In general, we focused on human-like behavior that could by analyzed by reading out information via the web browser, i.e., keyboard and mouse events.

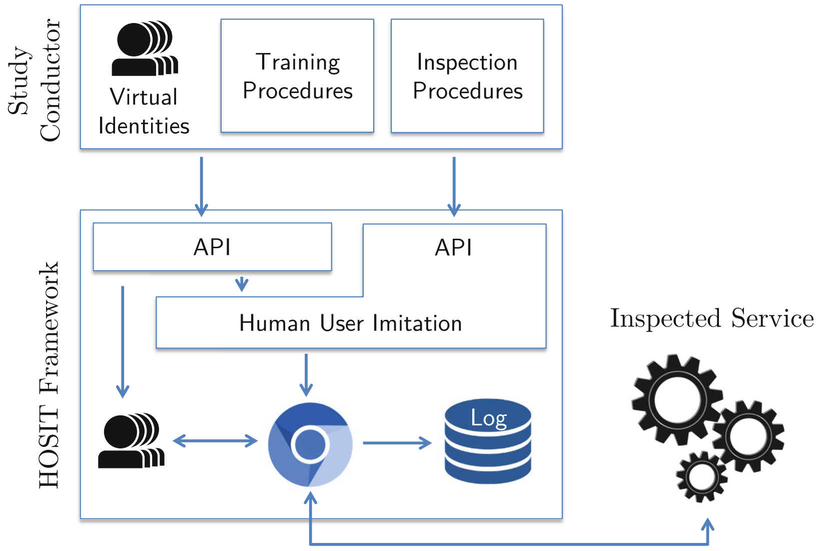


Fig. 1. Architecture of HOSIT

We considered the human-like behavior based on empirical studies modeling human computer interaction [8,14,25,33,37,46] as well as similar ideas on human behavior simulation [5].

The basic architecture of HOSIT is as follows (see Fig. 1). In order to test services, the study conductor creates one or multiple virtual identities with different browsing behavior (e.g., typing speed and clicking behavior). The conductor also defines a sequence of activities to be executed on the tested services for the respective study. Examples for activities can be: "click on *shopping cart* link", "search for a friend", or "logout from service". In many cases, these activities can be divided into training procedures (let the service learn "normal" user behavior) and inspection procedures (analyze the service's reaction to unusual behavior). The HOSIT API offers functions to create virtual identities inside the HOSIT framework as well as to execute the activities. In contrast to other solutions, HOSIT enables human-like behavior in two ways. First, human-imitating behavior is automatically added to activity calls for many functions (e.g., the "click on button" function clicks on an arbitrary position inside the button). And second, HOSIT offers additional function calls allowing explicit human behavior as a part of the activity sequence (e.g., "scroll to the end of the web page"). Using a script containing a sequence of activities and the browsing behavior from the virtual identities, the HOSIT framework calls the service using a Chromium browser instance. Finally, all responses from the service are logged for later analysis.

Figure 2 shows a simple example of a HOSIT script calling an online service in a human-imitating manner. The example code invokes HOSIT to open a search

```
// Open new page tab
await controller.newPage("https://www.startpage.com/");

// Wait a random time period
await controller.randomWait();

// Click on the "Images"-Link
await controller.click("a[href='https://www.startpage.com/
    en/pics.html']");

// Wait until the text field is loaded
await controller.waitForSelector("input[type='text']");

// Generate and enter search query based on
// current events in media
await controller.typeSearchQuery("input[type='text']");

// Scroll to the bottom of the page
await controller.scrollToBottom();
```

Fig. 2. Example HOSIT script

engine, click on the link to open the image search (after waiting a random period), and enter a search query chosen randomly by HOSIT based on current events in the media. Finally, HOSIT scrolls to the bottom of the page. This results in a usage of the online service in a way that a human would also do.

All activities performed on the online service as well as errors are logged into a database. As a result, study conductors get an overview of all interactions that the identities performed on the online service. This also eases the debugging of errors caused by activities of a certain identity.

4 Implementation

We implemented HOSIT using the Node.js library Puppeteer [21] in version 0.13.0 for browser automation. Consequently, HOSIT can be used on all operating systems which are capable of running the Node.js runtime environment and the browser Chromium. A Chromium version is bundled with the HOSIT installation. Nevertheless, HOSIT can be configured to use a customized Chromium or Chrome browser version instead of the bundled version. This might be necessary for example, when testing websites requiring Digital Rights Management (DRM) functionalities, which are included in Chrome but not in Chromium.

Chromium is executed in a custom *headful* mode, in which the browser is launched in the standard mode with visible GUI[3]. HOSIT uses this headful mode to minimize the detection of automated browsing. Chromium's headless

[3] To be compatible with Linux servers or Docker containers without a visible desktop environment, the headful mode can also be run inside a virtual window session.

Table 1. Feature differences between Puppeteer and HOSIT

Properties	Puppeteer 0.13.0	HOSIT
Typing speed	Constant	Randomized variations
Click position	Exact element center	Randomized variations
Click time	0 ms	Realistic [25]
Logging	Limited	Extended[a]
Browsing behavior changes	-	Yes, based on persona
Bot detection protection	-	Patched
Functions		
Common workflows	Need to be repeated	Integrated in Controller class
Search query generator	-	Included
CAPTCHA solving	-	Included
Scrolling	-	Included
Select tabs	-	Included

- Not included

[a]Logs all conducted actions with screenshots into a database

mode, which is designed specifically for browser automation, can be detected by a number of differences in the browser's properties and behavior [40]. During testing we actually experienced that online services treat headless browsers differently. Amazon, for example, required a CAPTCHA in headless but not in headful mode. We also patched HOSIT against known headless browser detection mechanisms [40]. HOSIT executes these patches when launching Chromium, e.g., removing the `navigator.webdriver` property [35].

During testing, we found some indications that browser automation can be detected with the standard functionality of Puppeteer. For instance, Amazon rated correctly entered CAPTCHA solutions as *not correct* if "typed" in by the standard Puppeteer function. Therefore, we enhanced some of Puppeteer's integrated functions with human-like user behavior. We compared manual browsing behavior with the automated behavior of Puppeteer to determine differences and optimized the affected functions. First, we modified the constant standard delays between pressing and releasing key buttons with randomized delays. These delays vary with an average typing speed which is defined by the identity (average time and maximum deviation). We recommend to measure these delays on real humans before setting them on the identities. By default, we set empirically measured typing speeds [8,14] on the identities. This procedure helped mimic human behavior more precisely. Further, we modified the mouse input behavior. Instead of clicking in the exact center of an element, the mouse selected a random click point in the center quarter of the element. We also replaced the default delay between pressing and releasing the left mouse button of 0 ms with an empirically measured clicking time with randomized variations [25].

We, moreover, added further functionalities to HOSIT which did not exist in Puppeteer (see Table 1). Finally, we simplified the API of Puppeteer and added recurrent tasks for the use case scenario inside the functions, e.g., automatically adjust the browser resolution when creating a new tab. As a result, fewer function calls are required to achieve the same result as with Puppeteer while being more human-like in many respects.

As stated in Sect. 3, each HOSIT instance is linked to a virtual identity which controls a browser instance. All further browsing behaviors are derived from this identity on this instance (e.g., typing behavior, selecting different categories based on the virtual identity's persona). The identity manages all browser tabs such as opening, switching, and closing browser tabs, and performs the actions on the website. These actions range from typing or clicking to scrolling and can only be performed in the currently open browser tab. We decided to select this identity-based structure to both optimize the API for the use case and to avoid unrealistic browsing behavior that was possible in Puppeteer, e.g., clicking buttons in two browser tabs at the same time.

When developing own studies of online services, study conductors have to design individual testing procedures with HOSIT. This is necessary since navigation structures and functionalities differ between online services and might change over time. For fine-grained variations of the browsing behavior, each HOSIT instance provides functions which can be used to increase randomized browsing behavior. These functions range from providing a random boolean value with a given probability for if-else conditions to providing the persona of the identity. By using these functions, we achieved that each browser session performed by HOSIT appeared differently on the tested online services.

5 Exemplary Use

To evaluate HOSIT, we conducted two studies that we discuss in the following. Both experiments would not have been possible without HOSIT or just with significant higher effort. The discussions will also provide a better understanding of HOSIT deployments based on the two given exemplary use case scenarios.

5.1 Use Case 1: RBA

Risk-based Authentication (RBA) [18] is an adaptive security measure to improve password authentication. During login, RBA monitors and stores additional features available in the context (e.g., IP address or user agent string) and requests additional information for authentication if a certain risk level is exceeded. RBA offers protection against security risks such as credential stuffing, password database leaks and intelligent password guessing methods. Beyond that, RBA has the potential to compensate low adoption rates of Two-factor Authentication (2FA). For instance, less than 10% of all active Google users activated 2FA in January 2018 [28].

RBA is recommended in the NIST digital identity guidelines [22] and is used by several large-scale online services. However, these online services keep their implementations secret and restrain their approaches for a public discussion in science.

This lack of public knowledge makes it difficult for small and medium websites to use RBA.

For this reason, we black box tested eight popular online services[4] with HOSIT to find out more about the corresponding RBA implementations, i.e., features and offered additional authentication factors [45]. We created 28 virtual online identities, registered 224 user accounts with the eight targeted services, and observed the services' behavior when accessing them under different circumstances. Each virtual identity had its own unique IP address from the same Internet service provider and a personal computer.

However, analyzing the inner workings of RBA is complicated, since one of the main tasks of RBA is to protect against bots. During pilot testing, we found indicators that some online services treated an automated browser using Puppeteer differently. For this reason, we designed our study using HOSIT to imitate human user behavior as exact as possible. Imitating human behavior was essential to make sure that the observed services' behavior is identical to normal usage.

RBA estimates the login risk based on the login history of the user. Therefore, our virtual identities conducted 20 browsing sessions including user sessions on the online services. The user sessions included login, activities on the online service, and logout. After these 20 browsing sessions, we varied browser features including the login time, IP address and device, logged in again on all online services, and observed the reactions. Based on the reactions, we drew conclusions about the inner RBA workings of the tested online services. The activities on the online services were randomized and individualized with HOSIT and differed on each of the online services. We selected typical activities for each of the online services, e.g., scrolling in the newsfeed, checking mail inbox or browsing for articles or jobs. In addition, these activities included a lot of randomness to mitigate being detected as a bot. As an example, on social media websites, it was randomly alternated between scrolling in the newsfeed, checking the message inbox and searching for content.

Since online services are likely tracking their users [7,9], all virtual identities simulated randomized browsing behavior in each browsing session with HOSIT. They visited search engines and entered search queries based on current topics discussed in media. Then, they opened some of the websites and "read" the text by scrolling and waiting. Also, the testing sequence of services was shuffled to a random order. This was done to prevent our virtual online identities from logging into the online services at similar times.

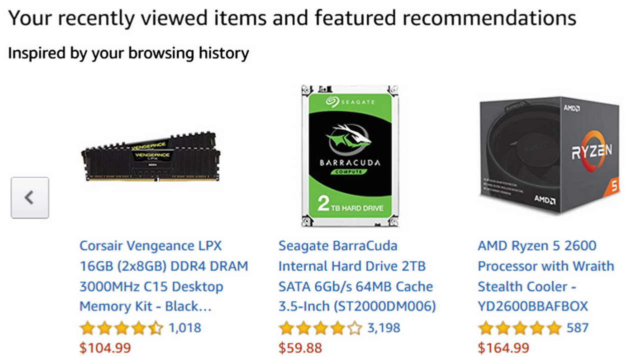
With the study based on HOSIT, we were able to derive features as well as an approximation to the respective weightings used for the RBA risk estimation of popular online services. One major finding was that five of the eight tested popular online services used RBA. Also, each of the services had a different RBA implementation, varying from protecting all users to only a selection of users. Besides using the IP address as a high weighted RBA feature, some services also used additional lower weighted features (e.g., user agent string).

More details on the RBA study can be found in the original publication [45].

[4] Amazon, Facebook, GOG.com, Google, iCloud, LinkedIn, Steam and Twitch.

5.2 Use Case 2: Amazon Product Recommendation System

When shopping on Amazon, a large amount of customer actions are tracked by the online shop. Besides the purchased items, these actions also include every item just visited by the user. Details can be seen in logs which European users can request from Amazon [3]. This right to request all personal data stored on a service provider is granted by the GDPR [16] (Fig. 3).



Your recently viewed items and featured recommendations

Inspired by your browsing history

Corsair Vengeance LPX 16GB (2x8GB) DDR4 DRAM 3000MHz C15 Desktop Memory Kit - Black... 1,018 $104.99

Seagate BarraCuda Internal Hard Drive 2TB SATA 6Gb/s 64MB Cache 3.5-Inch (ST2000DM006) 3,198 $59.88

AMD Ryzen 5 2600 Processor with Wraith Stealth Cooler - YD2600BBAFBOX 587 $164.99

Fig. 3. Shopping history and recommendations in the Amazon online shop

Amazon offers different types of product recommendations that are considered interesting for the customer [31]. When visiting a product page for example, similar or related items are presented. These items are based on sponsoring (*"Sponsored products related to this item"*) or shopping behavior of other users (*"Customers who bought this item also bought"*). Another recommendation type (*"Inspired by your browsing history"*) is based on the user's own browsing history mentioned in the previous paragraph, i.e., not only the purchase history, but also items just visited.

The recommendations given by Amazon are interesting for customers as well as other online shops. Hence, these recommendations can be considered a valuable asset for Amazon. It is therefore a reasonable assumption that Amazon, by detecting bots, is protecting these assets from automatic scraping. As a countermeasure, a bot could be presented different website content compared to human users, e.g., a CAPTCHA, different recommendations, or even recommendations with different prices. Thus, research on recommendations shown to human users requires a human-imitating client as provided by HOSIT.

In order to analyze the recommendation system and to verify this assumption, we conducted a study on the Amazon online shop. In this study, our (automated) user requested a fixed sequence of products and recorded the recommended products on the history page.

We conducted the same study with three different types of clients: automatically using Puppeteer, automatically using HOSIT, and manually by a human user. In addition, the products were requested in two different manners: either by simply opening the sequence of product page URLs or with "human like" online shopping, i.e., typing a search term into the search bar, selecting a search result, looking at the product page, searching for a next item and so forth. Finally, we performed this study with both registered and unregistered Amazon users.

The evaluation of the study revealed an unexpected result: the recommended items were exactly the same in all cases, including the order of items and the product prices. Thus, in contrast to the RBA of Amazon services, we assume that Amazon does not perform any bot detection for their recommendation system or allows bots to a certain degree, e.g., let harmless bots pass, block bots exaggerating the network traffic [1].

6 Benefits and Limitations

We put a lot of effort into ensuring that our tool was not recognized as a bot by online services. Nevertheless, the possibility that online services recognize HOSIT-based experiments as automated browsing remains. Even human-like browsing if performed constantly for a very long time will surely be detected. Also, creating too many new user accounts from the same IP address in a short time is likely to be noticed and even stopped by many online services. This, however, is even true when performed by a human. Thus, despite all protection mechanisms, automated browsing activities should not be exaggerated and kept at a realistic level, e.g., by introducing a long pause after some hours.

Still, based on our observations, we are convinced that our tool remained under respective bot detection thresholds. For instance, Amazon did not block automated logins with HOSIT while it did with Puppeteer. In March 2019, we also tested HOSIT using an instance of reCAPTCHA v3 [20], which is specifically designed to recognize bots. It analyzed the browsing behavior and returned a risk score. The score was a numerical value between 1.0 (*very likely a human*) and 0.0 (*very likely a bot*). We opened a testing website, which used reCAPTCHA v3, with both Puppeteer and HOSIT, and observed the risk score returned by the reCAPTCHA API. When using HOSIT, the reCAPTCHA v3 risk scores were identical to those of a human-controlled Chrome browser with empty browsing history and cookies (score: 0.7 = *likely a human*), while this was not the case with Puppeteer (score: 0.1 = *likely a bot*). After the release of HOSIT in April 2019, the reCAPTCHA risk score when using HOSIT was lowered to 0.3. This again underlines the arms race between bot detectors and bot detection avoiders. We will observe novel bot detection mechanisms and integrate countermeasures against them in future versions of HOSIT.

Before conducting research studies with HOSIT, study conductors are advised to test for anomalies on online services. In addition, study conductors should monitor which JavaScript attributes were read by online services while accessing this service [41]. These tests are helpful to determine possible bot detection and to implement countermeasures as a result.

Overall, we still find HOSIT highly sensible for studies due to the following reasons: (i) The reCAPTCHA v3 risk score is still higher than with Puppeteer. (ii) Not all online services on the Internet use the current reCAPTCHA. (iii) The API of HOSIT is more simplified than the API of comparable tools such as Selenium and Puppeteer, which makes it much easier to use.

7 Ethical Considerations

As with most tools for black box analysis, HOSIT is considered as "dual use", i.e., it can be used for illegitimate purposes as well. On the one hand, it can be beneficial to gather information on service behavior determining our everyday life. On the other hand, it could also be used for click fraud on online advertising, theft of intellectual property, or possibly even denial of service. Further, when using HOSIT, researchers should carefully check not to violate the respective *Terms of Service*. Also, researchers should use automated browsing responsibly, e.g., by keeping the impact on the inspected online services minimal [29,45].

We believe, however, that the results gathered by public research with HOSIT can be beneficial for a large user base and thus should be set ahead of corporate goals. We further argue that our work is justified, as the expected gain from scientific studies outweighs the potential security implications. Ultimately, we hope that public research based on our inspection tool will be beneficial for smaller online services. In consequence, security related research using this tool will protect a larger user base.

8 Conclusion

In this paper we presented HOSIT, a framework for automatically invoking online services in a human-like manner. As many online services try to detect if the client is a person or a bot, human-imitating behavior is required for automated service interactions in order to receive the same results as a human user. HOSIT implements a number of human-like behavior techniques and can be extended with further methods, as required by the targeted experiment and online service.

HOSIT can be used to circumvent services' bot-detection and to perform large-scale research on how online services behave towards human users. This is particularly interesting if the offered service depends—or is suspected to depend—on the user's behavior, location, history, device, and so on. Examples for such services are results from search engines, information in social networks, or recommendations in online shops. In particular, our research on RBA [45], which led to valuable and beneficial results, would not have even been possible without HOSIT. We discovered—among others—a privacy leakage in one of the RBA dialogs of Facebook and resolved this issue within a responsible disclosure process.

In future work, we will continuously extend and refine the human-imitating techniques of HOSIT. To evaluate their effectiveness, we will perform in-depth analysis on the influence of our methods on bot-detection systems, such as reCAPTCHA, on a regular basis. Moreover, we will apply HOSIT to study further scenarios including, e.g., search engine results and local browser storage usage patterns. We hope to see more of such research conducted on the basis of HOSIT.

For future research on service behavior, we will also follow alternative approaches. Instead of performing black box tests using camouflaged tools, services could enable responsible access to researchers. Researchers would benefit from unbiased results and focus on the analysis (and not on the black box testing tools), and services could advertise their support for research. This responsible service access could be monitored by an independent organization or public authority. A similar method called *regulatory sandbox* is used successfully in the financial area [38] and is currently discussed for research on personal identifying information [39].

Acknowledgements. We would like to thank Tanvi Patil for proofreading a draft of the paper. This research was supported by the research training group "Human Centered Systems Security" (NERD.NRW) sponsored by the state of North Rhine-Westphalia.

References

1. Akamai: Bot-Manager, January 2018. https://www.akamai.com/us/en/multimedia/documents/product-brief/bot-manager-product-brief.pdf
2. Allen, N.A.: Risk based authentication. Patent number US9202038B1 (2015)
3. Amazon: Amazon.co.uk Help: How do I request my data? (2019). https://www.amazon.co.uk/gp/help/customer/display.html?nodeId=G5NBVNN2RHXD5BUW
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
5. Blythe, J., et al.: Testing cyber security with simulated humans. In: IAAI 2011, San Francisco, CA, USA, August 2011
6. Bond, R.M., et al.: A 61-million-person experiment in social influence and political mobilization. Nature **489**(7415), 295–298 (2012)
7. Bujlow, T., Carela-Espanol, V., Lee, B.R., Barlet-Ros, P.: A survey on web tracking: mechanisms, implications, and defenses. Proc. IEEE **105**(8), 1476–1510 (2017)
8. Card, S.K., Moran, T.P., Newell, A.: The keystroke-level model for user performance time with interactive systems. Commun. ACM **23**(7), 396–410 (1980)
9. Chaabane, A., Kaafar, M.A., Boreli, R.: Big friend is watching you: analyzing online social networks tracking capabilities. In: WOSN 2012, Helsinki, Finland, pp. 7–12. ACM, August 2012
10. Choudhary, S.R., Prasad, M.R., Alessandro Orso: X-PERT: a web application testing tool for cross-browser inconsistency detection. In: ISSTA 2014, San Jose, CA, USA, pp. 417–420. ACM (2014)

11. Dalai, A.K., Jena, S.K.: Online identification of illegitimate web server requests. In: Venugopal, K.R., Patnaik, L.M. (eds.) ICIP 2011. CCIS, vol. 157, pp. 123–131. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22786-8_15
12. Degeling, M., Utz, C., Lentzsch, C., Hosseini, H., Schaub, F., Holz, T.: We value your privacy ... Now take some cookies: measuring the GDPR's impact on web privacy. In: NDSS 2019, San Diego, CA, USA, February 2019
13. DETER Project: DASH user guide (2014). https://deter-project.org/sites/deter-test.isi.edu/files/files/dash_users_guide.pdf
14. Drury, C.G., Hoffmann, E.R.: A model for movement time on data-entry keyboards. Ergonomics **35**(2), 129–147 (1992)
15. Englehardt, S., Narayanan, A.: Online tracking: a 1-million-site measurement and analysis. In: CCS 2016, Vienna, Austria, pp. 1388–1401. ACM, October 2016
16. European Parliament and Council: Regulation (EU) 2016/679 (GDPR), January 2016. http://data.europa.eu/eli/reg/2016/679/oj/eng
17. Franken, G., Goethem, T.V., Joosen, W.: Who left open the cookie jar? A comprehensive evaluation of third-party cookie policies. In: USENIX Security 2018, Baltimore, MD, USA, August 2018
18. Freeman, D., Jain, S., Duermuth, M., Biggio, B., Giacinto, G.: Who are you? A statistical approach to measuring user authenticity. In: NDSS 2016, San Diego, CA, USA, February 2016
19. Golla, M., Dürmuth, M.: On the accuracy of password strength meters. In: CCS 2018, Toronto, Canada, pp. 1567–1582. ACM, October 2018
20. Google: reCAPTCHA v3, July 2019. https://developers.google.com/recaptcha/docs/v3
21. Google Chrome: Puppeteer - Headless Chrome node API, July 2019. https://github.com/googlechrome/puppeteer
22. Grassi, P.A., et al.: Digital identity guidelines: authentication and lifecycle management. Technical report, NIST SP 800–63b, National Institute of Standards and Technology, Gaithersburg, MD, June 2017
23. Iaroshevych, O.: Improving second factor authentication challenges to help protect Facebook account owners. In: SOUPS 2017, Santa Clara, CA, USA. USENIX Association, July 2017
24. Judd, T., Kennedy, G.: A five-year study of on-campus Internet use by undergraduate biomedical students. Comput. Educ. **55**(4), 1564–1571 (2010)
25. Komandur, S., Johnson, P.W., Storch, R.: Relation between mouse button click duration and muscle contraction time. In: EMBC 2008. IEEE, August 2008
26. Li, T.-C., Hang, H., Faloutsos, M., Efstathopoulos, P.: TrackAdvisor: taking back browsing privacy from third-party trackers. In: Mirkovic, J., Liu, Y. (eds.) PAM 2015. LNCS, vol. 8995, pp. 277–289. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15509-8_21
27. Mark, G., Wang, Y., Niiya, M.: Stress and multitasking in everyday college life: an empirical study of online activity. In: CHI 2014, Toronto, Canada. ACM (2014)
28. Milka, G.: Anatomy of account takeover. In: Enigma 2018, Santa Clara, CA. USENIX Association, January 2018. https://www.usenix.org/node/208154
29. Petsas, T., Tsirantonakis, G., Athanasopoulos, E., Ioannidis, S.: Two-factor authentication: is the world ready?: quantifying 2FA adoption. In: EuroSec 2015, Bordeaux, France, pp. 4:1–4:7. ACM, April 2015
30. Rsmwe: Rakuten.com Chrome Headless Detection, February 2018. https://github.com/Rsmwe/Headless-detected-demo
31. Smith, B., Linden, G.: Two decades of recommender systems at Amazon.com. IEEE Internet Comput. **21**(3), 12–18 (2017)

32. Snickars, P., Mähler, R.: SpotiBot - turing testing spotify. Digit. Hum. Q. **12**, 12 (2018)
33. Soukoreff, R.W., MacKenzie, I.S.: Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. Int. J. Hum. Comput. Stud. **61**(6), 751–789 (2004)
34. Starov, O., Nikiforakis, N.: XHOUND: quantifying the fingerprintability of browser extensions. In: IEEE S&P, San Jose, CA, USA. IEEE, May 2017
35. Steward, S., Burns, D.: WebDriver - W3C Recommendation, June 2018. https://www.w3.org/TR/webdriver1/
36. Sulikowski, P., Zdziebko, T., Turzyński, D., Kańtoch, E.: Human-website interaction monitoring in recommender systems. Procedia Comput. Sci. **126**, 1587–1596 (2018)
37. Trauzettel-Klosinski, S., Dietz, K.: Standardized assessment of reading performance: the new international reading speed texts IReST. Investig. Opthalmol. Vis. Sci. **53**(9), 5452 (2012)
38. UK Financial Conduct Authority: Regulatory Sandbox Lessons Learned Report (2017). https://www.fca.org.uk/publication/research-and-data/regulatory-sandbox-lessons-learned-report.pdf
39. UK Information Commissioner's Office: Call for Views on Building a Sandbox: Summary of Responses and ICO Comment (2018). https://ico.org.uk/media/about-the-ico/consultations/2260322/201811-sandbox-call-for-views-analysis.pdf
40. Vastel, A.: Detecting Chrome headless, new techniques, January 2018. https://antoinevastel.com/bot%20detection/2018/01/17/detect-chrome-headless-v2.html
41. Vastel, A.: How to monitor the execution of JavaScript code with Puppeteer and Chrome headless, June 2019. https://antoinevastel.com/javascript/2019/06/10/monitor-js-execution.html
42. Velayathan, G., Yamada, S.: Behavior-based web page evaluation. In: WI-IAT 2006, pp. 409–412, December 2006
43. Venkatadri, G., Lucherini, E., Sapiezynski, P., Mislove, A.: Investigating sources of PII used in Facebook's targeted advertising. In: PETS 2019, pp. 227–244 (2019)
44. W3Schools: Browser Statistics: The Most Popular Browsers (2019). https://www.w3schools.com/browsers/default.asp
45. Wiefling, S., Lo Iacono, L., Dürmuth, M.: Is this really you? An empirical study on risk-based authentication applied in the wild. In: Dhillon, G., Karlsson, F., Hedström, K., Zúquete, A. (eds.) SEC 2019. IFIPAICT, vol. 562, pp. 134–148. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22312-0_10
46. Williams, J.L., Skinner, C.H., Floyd, R.G., Hale, A.D., Neddenriep, C., Kirk, E.P.: Words correct per minute: the variance in standardized reading scores accounted for by reading speed. Psychol. Sch. **48**(2), 87–101 (2011)

A Study of Security Vulnerabilities and Software Weaknesses in Vehicles

Wenjun Xiong(✉), Melek Gülsever, Koray Mustafa Kaya, and Robert Lagerström

School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden
{wenjx,gulsever,kkaya,robertl}@kth.se

Abstract. In this paper, we conduct an empirical study with the purpose of identifying common security vulnerabilities discovered in vehicles. The vulnerability information is gathered for 60 vehicle OEMs (Original Equipment Manufacturers) and common vehicle components from the National Vulnerability Database (NVD). Each vulnerability (CVE) is analyzed with respect to its software weakness type (CWE) and severity score (CVSS). 44 unique CVEs were found in NVD and analyzed. The analysis results show that about 50% of the vulnerabilities fall into the medium severity category, and the three most common software weaknesses reported are protection mechanism failure, buffer errors, and information disclosure.

Keywords: Vehicles · Cyber security · Vulnerabilities · Weaknesses

1 Introduction

Modern vehicles are often connected to the Internet, and they contain more than a hundred Electronic Control Units (ECUs) that control brakes, airbags, parts of the engine, and so forth. This combination of ECUs, sensors, and network buses creates a computerized system. The most commonly used network in a vehicle is called Controller Area Network (CAN)[1], and there are several known ways to breach this network [4]. Vehicles seem to be vulnerable to exploits in several ways, just as other systems are, but a malicious actor getting access to vital ECUs can have dire safety consequences. Vulnerabilities have been reported numerous times, and one famous example is when two white-hat hackers (penetration testers) acquired remote control of a 2014 Jeep Cherokee[2].

An empirical study of vehicle-specific vulnerabilities could help to improve the security of modern vehicles. We conduct this study by first collecting vulnerability data, using Common Vulnerability and Exposures (CVEs)[3], to get

[1] https://en.wikipedia.org/wiki/CAN_bus.
[2] https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.
[3] https://cve.mitre.org.

an overview. The vulnerability information for 60 Original Equipment Manufacturers (OEMs) including BMW, Mercedes-Benz, General Motors, and Tesla, together with common vehicle modules and network buses e.g. ECU, Bluetooth, Keyless Entry, and CAN bus was gathered from the National Vulnerability Database (NVD)[4]. We then analyzed the severity of these vulnerabilities through the Common Vulnerability Scoring System (CVSS)[5] ranking. Based on the results, we propose possible mitigations where OEMs can make efforts to improve security. NVD analysts have also integrated Common Weakness Enumeration Specifications (CWEs)[6] into the scoring process. The CWE system is used for categorizing and defining software weaknesses and provide well-documented information about these. This makes it possible for companies to publish and discuss vulnerabilities on a common platform where everyone has an understanding of what the definitions and terms represent. It also makes it easier to understand what a vulnerability specifically is and what weakness it is originating from.

In this paper, we follow these research steps: First, we gather vulnerability information through searching NVD. Then, we collect publicly available CVEs, the related CWEs, and CVSS information. After that, we analyze the identified vulnerabilities in terms of CVSS severity, CVSS metrics, and software weaknesses (CWEs), to understand the underlying causes and the impact of each vehicle-related vulnerability. 44 vehicle vulnerabilities are analyzed, and about 50% of them have medium severity. Based on the vulnerabilities found, 21 software weakness types are identified, where the most frequent one is protection mechanism failure.

The remainder of the paper is as follows. Section 2 considers the background. In Sect. 3, we explain the method used in this paper. Section 4 presents detailed steps and the analysis results. The paper is discussed in Sect. 5 and finally concluded in Sect. 6.

2 Background

The background section starts with an overview of vehicle security. Then we introduce the publicly available vulnerability sources, and finally, related work using vulnerability sources is presented.

2.1 Vehicle Security

Previously, vehicle OEMs did not consider cyber attacks that much, since an attack was only possible if an attacker had physical access to the vehicle. However, as modern vehicles have multiple wireless connections to outside networks and devices (e.g. Bluetooth, Internet), they are vulnerable to cyber attacks[7].

[4] https://nvd.nist.gov.

[5] https://www.first.org/cvss/.

[6] https://cwe.mitre.org.

[7] https://www.cpomagazine.com/cyber-security/connected-cars-a-new-and-dangerous-vector-for-cyber-attacks/.

A connected vehicle consists of over 100 ECUs, and each of them is responsible for one or more functionality. For example, the Telematics unit tracks vehicle diagnostics, driving behavior, location, and other information. Researchers got remote code execution on a Telematics unit of a vehicle by exploiting a vulnerability in the Bluetooth stack of an ECU and separately compromising a cellular modem [3].

The vehicle modules are connected through buses e.g. CAN bus, LIN (Local Interconnect Network) bus, which also have potential vulnerabilities. The work presented in [2] addressed the security issues in the CAN protocol, including lack of authentication, lack of network segmentation, lack of data encryption, and vulnerable to denial-of-service (DoS) attacks.

Possible security mechanisms to secure vehicle internal communication were addressed by (for instance) HoliSec[8], which include Message Authentication Codes (MAC) for traffic integrity, firewalls both for external traffic and for internal traffic implemented in gateway ECUs, use of Intrusion Detection Systems (IDSs) to detect unusual activities on the networks, and certificates for identification of various devices. Security mechanisms are also addressed in [2] for asset threat mitigation, including access control, packet filter firewall, message authentication, etc.

2.2 Publicly Available Vulnerability Sources

There are multiple vulnerability databases where vulnerabilities have been collected and are publicly available, such as NVD and SecurityFocus[9]. Also, many manufacturers regularly publish security advisories when they find vulnerabilities in their products.

In this study, data was gathered from NVD, which is the U.S. government repository for vulnerabilities. NVD uses the CVSS ranking system to distinguish between dangerous and less dangerous vulnerabilities. The CVSS scores in NVD has been proven to be trustworthy [9]. Furthermore, we use CVSS v2.0 in this study, as some identified CVEs do not have base metric scores of version 3.0 and they all have version 2.0 scores (more details can be found in Sect. 3.3). The base score represents the intrinsic and fundamental characteristics of a vulnerability that are constant over time and user environments.

2.3 Vulnerability Studies

There have been previous studies using publicly available vulnerability sources. In [9] the credibility of CVSS scores in five leading vulnerability databases was studied and the researchers found that CVSS is a robust system that can be trusted. It also provides a baseline for comparing results, i.e. average values of the CVSS base metrics. Also, the most accurate metrics in the databases seem to be the impact on Confidentiality, Integrity, and Availability (CIA).

[8] http://autosec.se/wp-content/uploads/2018/04/1.2-holisec-state-of-the-art.pdf.
[9] https://www.securityfocus.com.

As generating attack graphs is useful in showing the targets within systems. However, determining the attacker privileges corresponding to all vulnerabilities and continuing this effort as new vulnerabilities emerge is impractical and requires significant effort and time. To address this [1] defined an enhanced categorization of attacker privileges used for generating attacker privileges from the vulnerabilities in NVD in order to automate the generation of attack graphs.

In [12] the authors propose a method to assess security vulnerabilities of the installed and the latest software versions used based on the CVSS vulnerability scoring system, and then suggested whether a software version upgrade is needed.

The work presented in [5] focus on two sets of data - (1) the exposures of attacks on embedded systems published in security conferences and literature, and (2) the published vulnerabilities specific to embedded systems. The result of this work is a set of attack classification criteria that serves as a basis for an attack taxonomy. Also focused on embedded systems, [13] conducted an empirical study for identifying common software weaknesses by gathering data from online databases, including NVD and ICS-CERT[10]. The authors used CVSS scores and CWEs in relation to each CVE found.

Our study combines the keywords and analyses done in [7] and [11], both focusing on vulnerabilities in connected cars but with different datasets and approaches. Besides the database NVD considered in this study, [7] also considered a security incident repository called Upstream[11], and in [11] data was also gathered for Qualcomm[12].

3 Method

The goal of this study is to identify common vulnerabilities in connected vehicles systematically and quantitatively. The research steps for conducting this study are shown in Fig. 1 and described below.

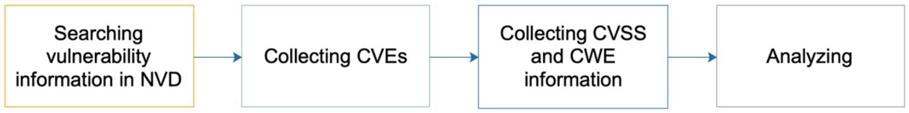

Searching vulnerability information in NVD
Collecting CVEs
Collecting CVSS and CWE information
Analyzing

Fig. 1. An overview of research steps.

- Step 1: searching for vulnerability information in NVD.
- Step 2: collecting publicly available vulnerabilities (CVEs) in NVD.
- Step 3: collecting CVSS base metrics and CWE information.
- Step 4: analyzing vulnerabilities in terms of CVSS severity, CVSS metrics, and weakness types (CWE).

[10] https://www.us-cert.gov/ics/advisories-by-vendor.
[11] https://www.upstream.auto/.
[12] https://www.qualcomm.com/products/snapdragon-820-automotive-platform.

3.1 Searching for Vulnerability Information in NVD

In the first step, we used three types of keywords:

(1) Vehicle-related terms to include all possible vulnerabilities and weaknesses related to connected vehicles.
(2) Common vehicle components and networks. See Table 1 for an overview of some vehicle components that could be found in the vulnerability sample set. Common vehicle bus systems[13] were also considered, including CAN, LIN, MOST, and FlexRay.
(3) Major OEMs. 60 vehicle OEMs were considered, of which 47 OEMs have no information available in the vulnerability database.

The complete list of keywords can be found below.

- `Part 1: Vehicle-Related Terms`
  vehicle, car, automotive.
- `Part 2: Common Vehicle Components and Networks`
  adaptive cruise control, adas, airbag, airbiquity, android auto, autoliv, bluetooth, braking system, carlink, carplay, collision prevention, control unit, cruise, drivesync, engine control, infotainment, keyless entry, lane keep assist, park assist, lidar, controller area network/CAN, local interconnect network/LIN, media oriented systems transport/MOST, flexray, OBD-II, aassive anti-theft system/PATS, radio data system, steering control, telematics, tire pressure/TPMS.
- `Part 3: Major OEMs`
  volkswagen, bmw, acura, audi, toyota, jeep, kia, mercedes-benz, skoda, renault, ford, peugeot, nissan, hyundai, opel, mazda, saab, seat, subaru, dacia, citroën, honda, great wall, cadillac, suzuki, land rover, rolls royce, porsche, jaguar, mitsubishi, chevrolet, mini, lexus, alfa romeo, lancia, snapdragon automobile, alpine, aston martin, bentley, bugatti, buick, changan, chrysler, daimler, dodge, dongfeng, ferrari, fiat, fisker, geely, general motors, infiniti, gmc, lamborghini, maserati, maclaren, tesla, pagani, ssangyong, tata motors.

3.2 Collecting Vehicle Vulnerabilities

During Step 2, 44 vulnerabilities with unique CVE numbers were identified in NVD. The identified 44 vulnerabilities describe vulnerabilities related to at least six different vehicle components and networks; keyless entry, Bluetooth, airbag, Telematics, and CAN network.

Several vulnerabilities are related to two OEMs - BMW (9) and Tesla (6). For BMW, all of the vulnerabilities identified are related to Infotainment component/Telematics control unit of BMW i Series, BMW X Series, BMW 3 Series, BMW 5 Series, and BMW 7 Series vehicles produced in 2012 through 2018,

[13] https://automotive.softing.com/en/standards/bus-systems.html.

Table 1. Examples of vehicle components covered.

Keyword	Reason	Examples
Airbag	An airbag is a vehicle occupant restraint system using a bag designed to inflate extremely quick then quickly deflate during a collision	Exploitation allows an attacker to send a crafted Unified Diagnostic Service (UDS) message to detonate the pyrotechnical charges; [6]
Keyless Entry	A keyless entry system is an electronic lock that controls access to a vehicle without using a traditional mechanical key	An keyless entry system makes Tesla Model S easier for attackers to clone a key fob within a few seconds
TCU	TCU (Telematics Control Unit) is an embedded system on board of a vehicle that controls the tracking of the vehicle	BMW vehicles produced in 2012 through 2018, allow a remote attack via a cellular network
Bluetooth	Bluetooth connectivity is a popular feature that allows an owner to pair their phone with their car	BMW i Series, BMW X Series, BMW 3 Series, BMW 5 Series, and BMW 7 Series vehicles produced in 2012 through 2018 allows a remote attack via Bluetooth when in pairing mode, leading to a Head Unit reboot
Gateway ECU	Gateway ECU is an ECU that connects two or more vehicular networks by acting as a bridge to them	Tesla Motors Model S automobile's Gateway ECU is susceptible to commands that may allow an attacker to install malicious software allowing the attacker to send messages to the vehicle's CAN bus
OBD-II	OBD-II monitors emissions, mileage, speed, and other useful data	BMW i Series, BMW X Series, BMW 3 Series, BMW 5 Series, and BMW 7 Series vehicles produced in 2012 through 2018 allows local attacks involving the USB or OBD-II interface

which opens to local/remote attacks. Whereas for Tesla, there were six vulnerabilities identified for different vehicle components, including entertainment system that triggers firmware code execution, Passive Keyless Entry and Start (PKES) system that allows an attacker to clone a key fob within a few seconds, and a Gateway ECU that allows an attacker to install malicious software and send messages to the vehicle's CAN bus.

3.3 Collecting CVSS and CWE Information

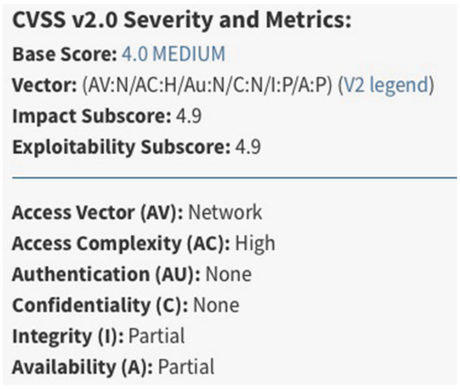
In this step, we collect CWE and CVSS metrics data for each CVE, as we aim to analyze the underlying causes of each vehicle-related vulnerability (CVE) as well as the impact of each vulnerability.

Table 2. Explanation of CVSS metrics that were studied.

Metric	Values	Description
Access Vector	Local	Local account or physical access required
	Adjacent	Access to either broadcast or collision domain
	Network	Remote access
Attack Complexity	High	Specialized access conditions
	Medium	Somewhat specialized
	Low	No specialized access conditions
Authentication	Multiple	Two or more times authenticated
	Single	Attacker needs to be logged on
	None	No authentication
Confidentiality	None	No impact
	Partial	Partial disclosure
	Complete	Total information disclosure
Integrity	None	No impact
	Partial	Modification of some files of data
	Complete	Total compromise of system integrity
Availability	None	No impact
	Partial	Reduced performance of interruptions
	Complete	Total shutdown of the affected resource

As an example of the first three research steps: (1) We use "Keyword Search" in NVD to collect vulnerabilities related to Gateway ECUs. (2) CVE-2016-9337 was identified, which addresses an issue discovered in Tesla Motors Model S automobile. All firmware versions before version 7.1 (2.36.31) with web browser functionality enabled makes the vehicle's Gateway ECU susceptible to commands that may allow an attacker to install malicious software allowing to send messages to the vehicle's CAN bus. (3) Its CVSS v2.0 severity and metrics information can be seen in Fig. 2. This CVE relates to the weakness type called - Command Injection (CWE-77).

16 out of 44 identified CVEs do not have base metric scores in CVSS version 3.0, however all of them have CVSS version 2.0 scores. Therefore, CVSS base metric scores of version 2.0 is used. Table 2 gives a short explanation of each of the CVSS version 2.0 metric, these will be used for further analysis.



CVSS v2.0 Severity and Metrics:

Base Score: 4.0 MEDIUM

Vector: (AV:N/AC:H/Au:N/C:N/I:P/A:P) (V2 legend)

Impact Subscore: 4.9

Exploitability Subscore: 4.9

Access Vector (AV): Network

Access Complexity (AC): High

Authentication (AU): None

Confidentiality (C): None

Integrity (I): Partial

Availability (A): Partial

Fig. 2. Example of a vulnerability entry, CVE-2016-9337, in NVD with its CVSS metrics.

37 out of 44 CVEs identified had a weakness number (CWE) associated with them and seven were found to be in the reserved status and thus missing CWE information in NVD (i.e. NVD-CWE-noinfo, NVD-CWE-Other). To be more specific, NVD-CWE-noinfo means there is insufficient information about the issue to classify it, and details are unknown or unspecified. NVD-CWE-Other means the weakness type is not covered by that subset. 21 different CWE weakness types were identified and will be further analyzed.

4 Analysis

In this section, we summarize and interpret the collected CVE data. Focusing on the CVSS metrics and the relations to software weaknesses (CWEs) that help describe the underlying causes of the vulnerabilities.

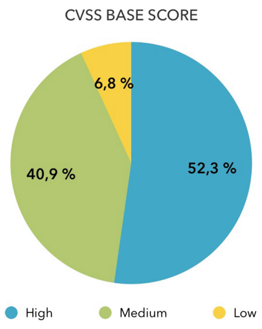


Fig. 3. CVSS severity base score distribution.

4.1 CVSS Metrics Analysis

First, we present the vulnerability analysis according to CVSS metrics (defined in Table 2). An overview of CVSS severity base score distribution is shown in Fig. 3. Value distributions of the three metrics, which capture how a vulnerability is accessed and whether or not extra conditions are required, are shown in Fig. 4; value distributions of the three impact metrics are shown in Fig. 5.

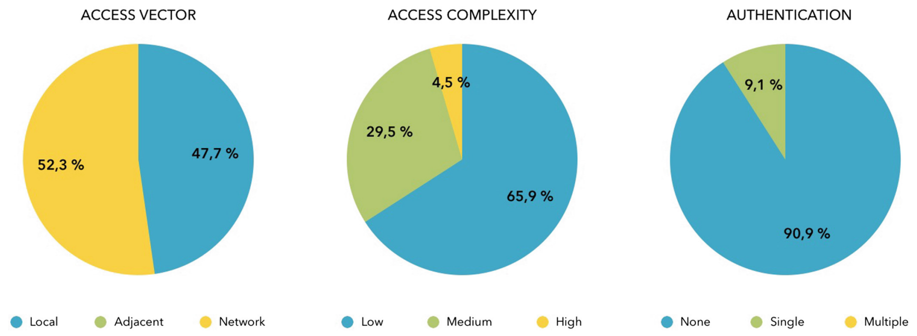


Fig. 4. CVSS metric value distributions for capturing how the vulnerability is accessed and whether extra conditions are required.

CVSS Base Score. The base score value ranges from 1 to 10, where medium vulnerabilities range from 4 to 6.9, and with 10 being the most severe. The score is based on six metrics, where Access Vector, Access Complexity, and Authentication metrics capture how a vulnerability is accessed and if extra conditions are required to exploit it, while Confidentiality, Integrity, and Availability (CIA) Impacts are defined as the degree of loss of CIA. These metrics are independent, for example, a vulnerability could cause a complete loss of integrity and availability, but no loss of confidentiality.

The results show that more than half (23) of the vulnerabilities fall into the high severity category (base score higher than 7) and only 3 vulnerabilities fall into the low severity category (e.g. CVE-2010-4565). Also, 18 vehicle vulnerabilities identified fall into the medium severity category.

Access Vector. This metric reflects the context by which vulnerability exploitation is possible. According to the analysis results shown in Fig. 4, 52% of the vulnerabilities can be exploited remotely. It is reasonable as vehicles are becoming Internet-facing systems, and are interconnected thus open to remote attacks. Whereas, none of these vulnerabilities belong to the adjacent network category, meaning the vulnerabilities are either remotely exploitable or they require attackers to access the target system locally.

Attack Complexity. This metric describes the conditions beyond the attacker's control that must exist to exploit the vulnerability. According to our analysis (shown in Fig. 4), 66% of the vulnerabilities can be exploited without restrictions. Only 2 out of 44 vulnerabilities require specialized access conditions.

Authentication. This metric measures the number of times an attacker must authenticate a target to exploit a vulnerability. As we can see in Fig. 4, almost all of the vulnerabilities (40 out of 44) require no authentication, while only 9% of the vulnerabilities require an attacker to be logged on, and none of the vulnerabilities need an attacker to be two or more times authenticated.

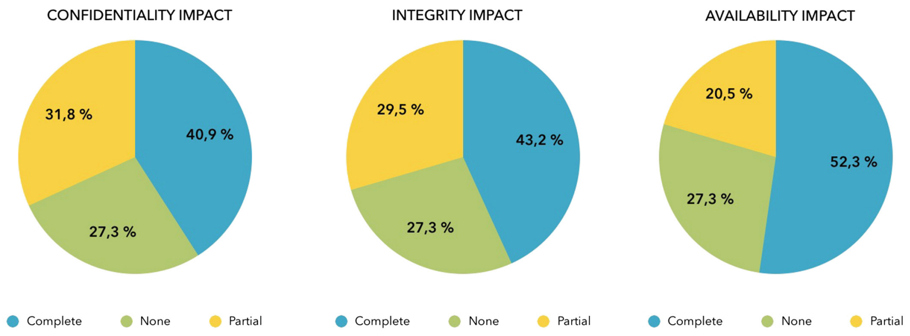


Fig. 5. CVSS base metric value distributions for CIA.

Confidentiality Impact. This metric measures the impact on confidentiality of a successfully exploited vulnerability. In our analysis, the plurality (41%) of these vulnerabilities lead to total information disclosure of vehicles, and only 27% of them have no impact on confidentiality. However, even when one vulnerability has total disclosure, it may not have a serious impact on connected vehicles unless it also leads to serious integrity and availability compromises.

Integrity Impact. Integrity Impact measures the trustworthiness and veracity of information, thus it is an important property for connected vehicles, and the partial and complete impact should be minimized. However, according to our analysis, the plurality (43%) of the vulnerabilities lead to full compromise.

Availability Impact. Availability Impact measures the impact a vulnerability can have on the accessibility of its information resources, such as a network service (e.g. web, database, email). According to the analysis, more than half of the vulnerabilities can fully access vulnerable resources. It is interesting to see that the portions of vulnerabilities that have no impact on Confidentiality, Integrity, and Availability are the same (27%), while not from the same set of vulnerabilities. This is worrisome as many connected vehicles have real-time constraints, therefore an Availability Impact with the Complete value can have serious consequences.

4.2 CWE Analysis

In this subsection, we further analyze the identified vulnerabilities through weakness types (CWEs). Based on the results, altogether 21 unique CWE types were found. As we can see in Fig. 6, where the horizontal axis denotes the occurrence frequency of a CWE, more than half of the CWEs occurred only once, and the most frequent CWE was found seven times. According to the descriptive statistics, the mean occurrence of a CWE was 1.76, and six of the CWEs belong to the top 25% occurrence group. We also employ the CVSS v2.0 score to understand the severity of the vulnerabilities for each CWE type.

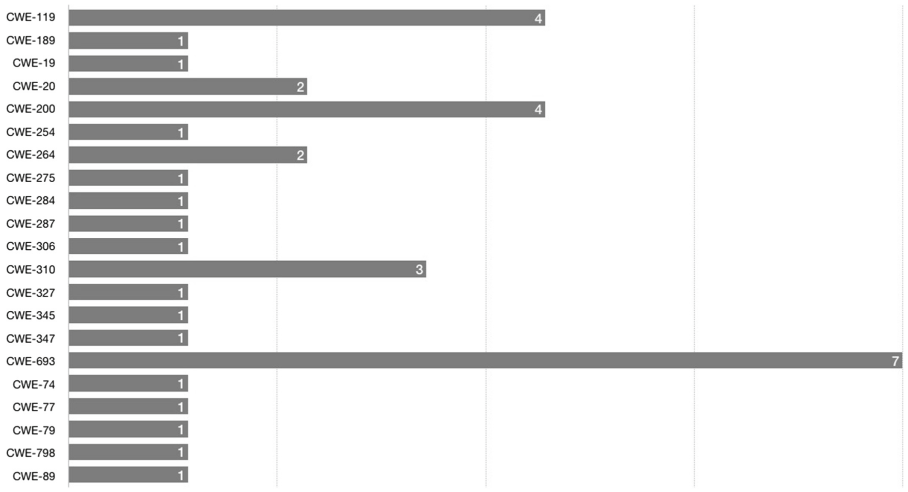


Fig. 6. Frequency of weakness types identified in this study.

As we can see in Fig. 7, six out of seven vulnerabilities belonging to CWE-693 (i.e. Protection Mechanism Failure) are ranked high, as these vulnerabilities lead to total compromise of both Confidentiality, Integrity, and Availability (CIA), whereas none of the vulnerabilities related to CWE-200 (i.e. Information Leak/Disclosure) and CWE-310 (i.e. Cryptographic Issues) have high severity.

Table 3 gives a short explanation of the top 25% CWE types identified, and provides suggested mitigations to counter these attack patterns (seen in Table 3).

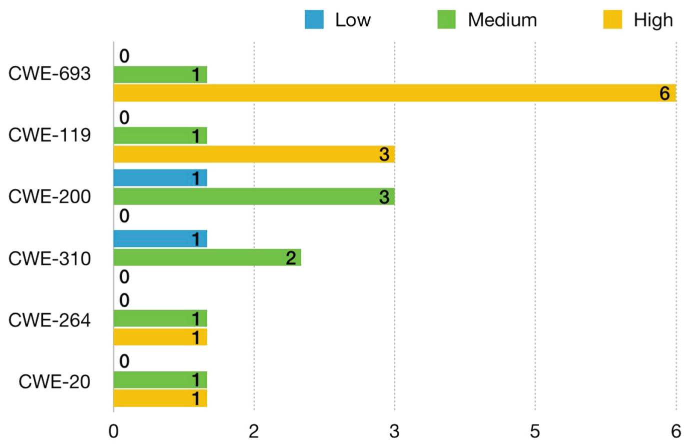


Fig. 7. Distribution of vulnerability severity of the top 25% common CWEs identified in this study.

Table 3. Top 25% of CWE types identified with possible mitigations.

CWE ID	Count	Description	Mitigations
CWE-693	7	Protection mechanism failure	Use intrusion detection systems; secure communications between ECUs
CWE-119	4	Buffer errors	Use a language, or a vetted library, or a framework that does not allow this weakness to occur; use static analysis tools
CWE-200	4	Information leak/disclosure	Set trust boundaries. Use privacy mechanisms
CWE-310	3	Cryptographic issues	Use strong cryptography and properly manage the private key
CWE-264	2	Permissions, privileges, and access control	Design secure architecture
CWE-20	2	Input validation	Use static and dynamic analysis tools to test the software

5 Discussion

In this paper, we conduct a study to find the most common weaknesses and vulnerabilities for connected vehicles and provide some possible mitigations to the most common weaknesses found. The top 25% of the weakness types found can be compared to the OWASP top 10 most critical web application security risks[14]. To be more specific, protection mechanism failure can be compared to broken authentication[15], which is the most critical security risk. Information leak/disclosure can be compared to sensitive data exposure[16]. Permissions, privileges, and access control can be compared to broken access control[17], and input validation can be compared to security misconfiguration[18]. Thus, our results overlap with the OWASP top 10.

The combination of ECUs, sensors, and network buses creates a computerized system in modern vehicles, which affects the potential number of vulnerabilities as each of these components might have weaknesses. The results show that 52% of the identified vulnerabilities can be remotely exploited (see in Fig. 4). It should also be noted that the vulnerabilities are not always isolated in exploits, as low-level vulnerabilities can be combined to achieve high-level attack goals [8]. Besides, 66% of the vulnerabilities can be exploited without specialized access conditions.

The OBD-II port was made mandatory for all cars in the U.S. 1996 and was introduced to all gasoline-fueled vehicles in Europe in 2001. During the study, we found that OBD-II related components have the largest number of vulnerabilities among the considered vehicle components, for example, it could allow remote attacks to execute arbitrary code by specifying an update server. If an OBD-II device has been compromised, its network connection can be the second point of vulnerability. Therefore, it is important to use security tools e.g. OBD-II scanners to diagnose and fix security problems.

The empirical study presented in [13] focusing on embedded systems in power networks identified the most common problem as improper input validation, whereas the most common vehicle weakness found in this study belongs to protection mechanism failure category. It implies that the protection mechanisms e.g. correct access control of vehicles may be missing, therefore, using intrusion detection systems and secure communications are important. In second place was buffer errors, also information leak/disclosure that leads to compromise of confidentiality is frequent, therefore, limit access to identifiable information, and use privacy mechanisms can help to increase the level of confidentiality [16].

Our study has its limitations. First, we only use information from the vulnerability database NVD, while there are other vulnerability sources e.g. Upstream for vehicle cyber attacks, ICS-CERT vendor advisories. Also, 60 major OEMs are considered in this study, however, most of them do not publish security advisories frequently. Due to the limited information available from NVD, the results are skewed towards BMW and Tesla. Furthermore, we only try to find the vulnerabilities of in-vehicle networks (e.g. CAN bus, FlexRay), and the attacks that

[14] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project.
[15] https://www.owasp.org/index.php/Top_10-2017_A2-Broken_Authentication.
[16] https://www.owasp.org/index.php/Top_10-2017_A3-Sensitive_Data_Exposure.
[17] https://www.owasp.org/index.php/Top_10-2017_A5-Broken_Access_Control.
[18] https://www.owasp.org/index.php/Top_10-2017_A6-Security_Misconfiguration.

can happen to them and to the connected components (e.g. ECUs). However, vehicular Ethernet networks, diagnostics protocols, some popular public interfaces, and advanced connectivity capabilities e.g. V2X communications are not included in this study.

This study can benefit our research. Threat modeling is proposed as a solution for secure application development and system security evaluations. It aims to be more proactive and make it more difficult for attackers to accomplish their malicious intents [15], and often, it is combined with attack simulations for providing probabilistic simulation results. Previously, we conducted threat modeling and attack simulations of connected vehicles [10,14] and found that more research in vehicle-specific attacks and countermeasures is needed. Thus, this work studying vehicle-specific vulnerabilities, weaknesses, and countermeasures can help to provide more accurate simulation inputs.

This study can also add value to the vehicle industry for producing more secured vehicles by identifying weakness types and providing their possible mitigations. As the connected vehicle is still in its early stage, a more secure vehicle architecture is needed. Future work includes conducting research focusing on the life cycle of modern vehicles, and further investigation of risk-prone industry standards.

6 Conclusion

In this study, we investigated security vulnerabilities in vehicles including their severity and impact, as well as the relation to software weaknesses. The information was identified and collected through the National Vulnerability Database (NVD). We identified several common weaknesses types. With the most common being protection mechanism failure, that may lead to a complete compromise of Confidentiality, Integrity, and Availability (CIA). As the vehicle industry needs to continuously work on cyber security as an integral part of product development, maintenance, and vehicle architecture design our study can provide some guidelines in this work.

References

1. Aksu, M., Bicakci, K., Dilek, M., Ozbayoglu, A., Tatlı, E.: Automated generation of attack graphs using nvd. In: CODASPY 2018 - Proceedings of the 8th ACM Conference on Data and Application Security and Privacy, pp. 135–142. Association for Computing Machinery (2018)
2. Buttigieg, R., Farrugia, M., Meli, C.: Security issues in controller area networks in automobiles. In: 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, pp. 1–6 (2017)
3. Checkoway, S., et al.: Comprehensive experimental analyses of automotive attack surfaces. In: USENIX Security Symposium, San Francisco, pp. 77–92 (2011)
4. Currie, R.: Hacking the can bus: basic manipulation of a modern automobile through can bus reverse engineering. The SANS Institute, InfoSec Reading Room Report Series (2017)

5. Dorottya Papp, Z.M., Buttyan, L.: Embedded systems security: threats, vulnerabilities, and attack taxonomy. In: 2015 13th Annual Conference on Privacy, Security and Trust (PST), pp. 145–152 (2015)
6. Durrwang, J., Braun, J., Rumez, M., Kriesten, R.: Security evaluation of an Airbag-ECU by reusing threat modeling artefacts. In: 2017 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 37–43. IEEE (2017)
7. Gülsever, M.: A Study on Vulnerabilities in Connected Cars. Degree project, KTH Royal Institute of Technology, Stockholm, Sweden (2019)
8. Jajodia, S.: Topological analysis of network attack vulnerability. In: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, p. 2. ACM (2007)
9. Johnson, P., Lagerström, R., Ekstedt, M., Franke, U.: Can the common vulnerability scoring system be trusted? A Bayesian analysis. IEEE Trans. Dependable Secur. Comput. **15**(6), 1002–1015 (2018)
10. Katsikeas, S., Johnson, P., Hacks, S., Lagerström, R.: Probabilistic modeling and simulation of vehicular cyber attacks: an application of the meta attack language. In: Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP) (2019)
11. Kaya, K.M.: A Study of Vulnerabilities and Weaknesses in Connected Cars. Degree project, KTH Royal Institute of Technology, Stockholm, Sweden (2019)
12. Treetippayaruk, S., Senivongse, T.: Security vulnerability assessment for software version upgrade. In: 2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 283–289 (2017)
13. Välja, M., Korman, M., Lagerström, R.: A study on software vulnerabilities and weaknesses of embedded systems in power networks. In: Proceedings of the 2nd Workshop on Cyber-Physical Security and Resilience in Smart Grids, pp. 47–52. ACM (2017)
14. Xiong, W., Krantz, F., Lagerström, R.: Threat modeling and attack simulations of connected vehicles: a research outlook. In: Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP) (2019)
15. Xiong, W., Lagerström, R.: Threat modeling - a systematic literature review. Comput. Secur. **84**, 53–69 (2019)
16. Xiong, W., Lagerström, R.: Threat modeling of connected vehicles: a privacy analysis and extension of vehicleLang. In: International Conference on Cyber Incident Response, Coordination, Containment & Control (Cyber Incident). IEEE (2019)

System and Software Security

Experimental Analysis of the Laser-Induced Instruction Skip Fault Model

Jean-Max Dutertre(✉), Timothé Riom, Olivier Potin, and Jean-Baptiste Rigaud

Mines Saint-Etienne, CEA-Tech, Centre CMP, 13541 Gardanne, France
{dutertre,riom,potin,rigaud}@emse.fr

Abstract. Microcontrollers storing valuable data or using security functions are vulnerable to fault injection attacks. Among the various types of faults, instruction skips induced at runtime proved to be effective against identification routines or encryption algorithms. Several research works assessed a fault model that consists in a single instruction skip, i.e. the ability to prevent one chosen instruction in a program from being executed. This assessment is used to design countermeasures able to withstand a single instruction skip. We question this fault model on experimental basis and report the possibility to induce with a laser an arbitrary number of instruction skips. This ability to erase entire sections of a firmware has strong implications regarding the design of countermeasures.

1 Introduction

Fault attacks (FA) consist in disturbing the operations of a target integrated circuit (IC) for the purpose of extracting secret information it may contain. Faults, or computation errors, are injected by means of altering the target environmental conditions [2] (e.g. its voltage, temperature, frequency, etc.). The induced information leakage generally aims at extracting a cryptographic key [4] or at providing an unauthorized access to some of the target functionalities [9].

Laser illumination may also be used to inject faults into an IC [7,16]. Laser is probably the most expensive fault injection means, however it makes it possible to inject faults with high accuracy even at advanced technology nodes [10]. It is accurate both in terms of timing (faults may be injected with laser pulses as short as a few picoseconds [12]) and in terms of location (it affects mainly the logic gate located within its spot size that may be as low as a few micrometers [7]). This explains why significant research work is dedicated to the study of laser-induced FAs.

The properties of the faults induced by laser (or by any other fault injection means) are referred to as a fault model (FM). It is often linked to a given attack scheme and expressed as an ability to meet requirements in terms of synchronization with the target activity and extension of the induced fault (e.g. the FM

A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 221–237, 2019.
https://doi.org/10.1007/978-3-030-35055-0_14

of the well-known Piret FA [14] requires to fault one byte of the AES algorithm calculations before its last MixColumn transformation).

In this work, we report our analysis of the laser-induced instruction skip FM. This FM relates to how a given instruction of a microcontroller program may be skipped (i.e. not executed) at runtime. Several works already described this FM and assessed the possibility of laser-induced single instruction skips [5,6,17]. The assessment of FMs on experimental grounds is of high interest regarding how FAs countermeasures (CMs) are designed and tailored. As a matter of example, the authors of [13] discussed two CMs based on instruction redundancy designed on the assumption that an attacker is only able to induce single instruction skips. Would this assumption be proved wrong, their CMs would be vulnerable.

Our experiments extend further this FM by reporting the feasibility of inducing several successive instruction skips by laser illumination. It also assesses the ability to skip several, but close, separate groups of instructions. This is a very strong FM which is very difficult to defend against with software only CMs.

This article is organized as follows. Section 2 discusses the state-of-the-art of instruction skips and introduces the aim of our work. Our experimental setup and settings are described in Sect. 3. Section 4 reports the obtained results. Then, the assessed FM is discussed in Sect. 5. Section 6 concludes the paper.

2 The Laser-Induced Instruction Skip Fault Model

2.1 Fault Model Definition

A FM generally describes the main properties of a FA scheme, often expressed in terms of requirements of synchronization (requirement to fault a particular step of an algorithm or program) and of extension (requirement to limit the fault extension, e.g. to a single bit or a single byte). In this work, we consider the FM related to laser fault injection. It may be defined at different levels of abstraction from transistor or gate level (as the authors of [10] did to describe laser-induced bit-set and bit-reset faults) to the assembler or algorithm level. We studied the FM of microcontrollers experiencing laser-induced instruction skips.

2.2 The Instruction Skip Fault Model, State-of-the-Art

An instruction skip is a fault that results in skipping, meaning not executing, one instruction of a microcontroller program at runtime (as if the program flow had skipped over the faulted instruction).

Several works studied the EM-induced instruction skip FM. Most of them assessed single instruction skips (in the same 8-bit microcontroller we used as target for [3], and on a 32-bit microcontroller for [13]). To the best of our knowledge, the only works reporting several successive instruction skips are [15] and [20]. [15] assessed four successive skips of instructions stored in the target instruction cache while [20] succeeded in faulting instructions stored in the target's pipeline.

Several works also deal with laser-induced instruction skips. [5,6] obtained single instruction skips with high accuracy and high success rate (on the same microcontroller we studied) and used it to perform a successful differential fault attack on AES. Still on the same target, [11] reports instruction skips based on resetting one or two bits of the targeted instruction opcode. The authors of [17] induced instruction skips on a more complex 32-bit cortex-M3 microcontroller. They were able to inject two single instruction skips distant from 58 ms to defeat a protected CRT-RSA algorithm. In terms of target complexity, [19] reports injection of single instruction skips into a quad core ARM cortex A9 microprocessor running at 1.4 GHz clock frequency. Hence, the state-of-the-art in laser-induced instruction skip was limited to single instruction skips (with a repetition rate in the range of tens of ms).

2.3 Study of Laser-Induced Instruction Skips

There is to date very few explanations of how an instruction skip is induced at the gate level, with the notable exception of [1]. It describes how increasing progressively the stress applied by a clock glitch to a microcontroller induces an increasing number of bit-reset faults into an instruction opcode. It results in (1) instruction modification at low stress or (2) in turning the instruction into an actual `nop` at high stress (i.e. the `no operation` instruction). Instruction modification achieves an instruction skip if the modified instruction has no effect on the code operations (instruction skips are often actual code modification); the same is true for turning an instruction into a `nop`. An analysis of single instruction skips due to instruction modification induced by laser is reported in [8]. It relates how faulting one bit of an instruction opcode led to two successful FAs.

Our research objective was to reproduce laser-induced instruction skips on a microcontroller and to study the main characteristics of its FM: accuracy, extent, success rate, time between successful skips, etc. Our aim was also to assess whether the single instruction skip fault model could be extended further to multiple instruction skips. This latter aim was of interest because CMs are based on the known FMs. Hence, a CM based on a too narrow FM may reveal vulnerabilities at test time. From previous experiments and taking into account the results of [1,5,6], we focused our experiments toward achieving instruction skips by turning the target instructions into `nop` instructions. Our experiments were carried out with the same 8-bit microcontroller studied by [1,3,5,6,11] for comparison purposes and also to ease the analysis of the obtained results.

3 Experimental Settings

3.1 Laser Bench

Laser Source Parameters. Our laser source is a nanosecond range laser source able to output a laser pulse with a 50 ns to 1 s tunable duration. It has a latency of less than 300 ns (i.e. the time interval between the trigger signal and the moment an actual laser pulse hits the target). Its wavelength is 1,064 nm (or

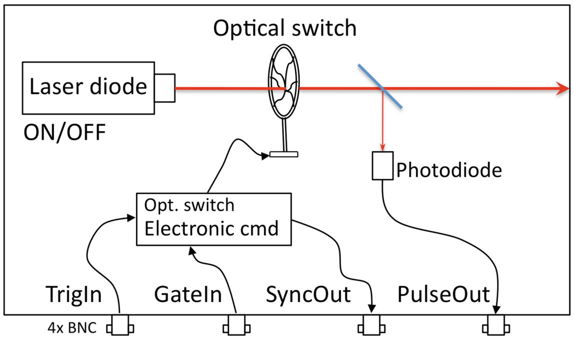


Fig. 1. Schematic of the main components of the used IR laser source.

near infrared, NIR). The use of a NIR laser source allowed us to perform our laser fault injection experiments through the target's rear side because it is able to pass through several hundreds of micrometers of silicon without being totally attenuated (the silicon die was also thinned down from 500 µm to 300 μm). The laser source max power is 3 W (measured at the fiber optic output) which is enough to inject faults into the target. We performed our experiments with a ×20 objective lens: it outputs a laser spot diameter of 5 µm. An infrared camera was used to adjust the laser spot focus and location w.r.t. the target's layout.

Laser Source Technology. Figure 1 provides a schematic view of the laser source technology. Any outputted laser pulse is carved from a continuous laser beam produced by a NIR laser diode thanks to an electro-optical switch (the laser diode operates on an ON/OFF basis). It takes 25 ns to open (resp. close) the optical switch of our source and to output a laser pulse that reaches the 3 W nominal power (resp. to extinguish a laser pulse). This explains that the shortest achievable laser pulse is 50 ns long. It also offers the ability to generate several consecutive laser pulses with a pause time in between as short as 50 ns. This repetition rate of 50 ns is a useful property for an attacker as underlined in Subsect. 4.4. An electronic board drives the optical switch (denoted Electronic cmd in Fig. 1). It features two laser shot modes, (1) the Trigger Mode and (2) the GateIn Mode:

- in Trigger Mode, a single laser pulse is produced in response to a voltage pulse delivered on the input BNC connector `TrigIn` (its duration, its power, and its delay w.r.t. the trigger signal are programmable),
- in GateIn Mode, the optical switch openings and closings follow the shape of the voltage signal applied on the input BNC connector `GateIn`. It makes it possible to generate a succession of voltage pulses of arbitrary shape (only constrained by the 25 ns open/close time of the switch).

A beam splitter captures a small amount of the laser pulse power that is converted into a voltage by a photodiode (see Fig. 1). It may be observed on the source `PulseOut` output BNC connector. This feature is useful to observe the synchronization of the actual laser pulse with the target activity.

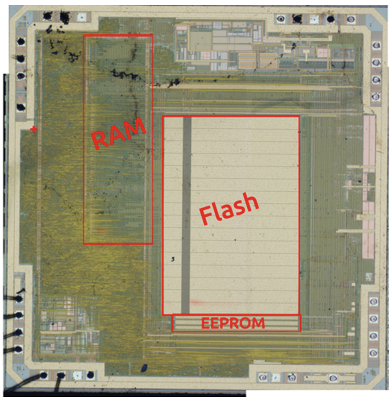


Fig. 2. Frontside view of the ATmega328P test chip - Flash, RAM and EEPROM memories highlighted in red. (Color figure online)

Laser Bench. The target is placed on a XYZ stage (with 0.1 µm accuracy). An IR camera is used to observe the location of the laser spot. The activity of the target and the laser source signals are recorded with an oscilloscope.

Laser-Sensitivity Map Drawing Process. Using a control PC to automate the process, we moved the laser over the area of the targeted device by displacement steps from 50 µm to 5 µm. For each position, we shot the laser while the microcontroller was running the test codes described in Subsect. 3.3. This allowed us to draw XY maps of laser-sensitivity: for each position where a fault was recorded, we drew a dot colored according to the obtained faults. Such laser-sensitivity maps were drawn for various laser power and timing.

3.2 Test Chip

We chose a simple target for the purpose of being able to analyse easily its answers to fault injection: an 8-bit non-secure ATmega328P microcontroller designed in the old CMOS 0.35 µm technology. It has 2 kB RAM, 3 kB Flash and 1 kB EEPROM memories; a Harvard architecture with a 2-stage fetch-execute pipeline. It runs at 16 MHz and has 32 general purpose registers. Registers r16 to r25 were used during our experiments. Figure 2 gives a front view of the test chip with its Flash, RAM and EEPROM memories highlighted in red. A red cross near the device bonding pads on its left shows the XY origin we used as a reference.

3.3 Test Codes

We studied the effect of laser-induced faults on dedicated test codes mostly written in assembly language. Our intent was to induce and analyze instruction skips by examining their effect on the assembly instructions of the test codes.

For each test series, we used two trigger signals for synchronization purposes (two outputs of the test chip):

Listing 1 Test code - Instruction skip analysis.

```
# Store 0x39 to 0x30 in RAM at address Z
# Initialize r16 to r25 at 0x55
# Set synchronization trigger
nop # 400 ns
# Set core trigger
ld r16,Z+      ld r16,Z+
ld r17,Z+      ld r17,Z+
ld r18,Z+      ld r18,Z+
ld r19,Z+      nop
ld r20,Z+      ld r20,Z+
ld r21,Z+      ld r21,Z+
ld r22,Z+      ld r22,Z+
ld r23,Z+      ld r23,Z+
ld r24,Z+      ld r24,Z+
ld r25,Z+      ld r25,Z+
# Clear core trigger
nop # 700 ns
# Clear synchronization trigger
# read back r16 to r25
```

– a synchronization trigger signal to accommodate for the latency of the laser source,
– a core trigger signal to synchronize the actual laser shot (thanks to the `PulseOut` signal) with the part of the assembly code of interest.

Listing 1 provides a description of the test code we used to tune our settings in order to induce instruction skips. The core part of the test code (encompassed by the core trigger) is a series of ten `ld rX,Z+` instructions, each one corresponding to a load in a destination register `rX` of a byte value stored in RAM memory at address `Z` with a post increment of `Z`. Prior to that, the ten destination registers, `r16` to `r25`, are initialized at `0x55` and an array of ten byte values `0x39` to `0x30` are stored in RAM with `Z` storing the address of its first element. Registers `r16` to `r25` are read back after the synchronization trigger is reseted (the two top blue signals in Fig. 4 are the synchronization and core triggers drawn for a fault free execution). The top part of Table 1 displays the values read back from `r16` to `r25` for a fault-free execution.

Table 1. Registers r16 to r25 readback values, for a fault free execution (top) and for an instruction skip targeting r19 (bottom, highlighted in red).

Register	16	17	18	19	20	21	22	23	24	25
Fault free	0x39	0x38	0x37	0x36	0x35	0x34	0x33	0x32	0x31	0x30

Faulted	0x39	0x38	0x37	0x55	0x36	0x35	0x34	0x33	0x32	0x31

As an example, the right column of the test code core part in Listing 1 displays the effect of a laser shot turning the `ld` instruction of line 9 into a `nop` instruction. The effect of such a laser-induced instruction skip is highlighted in the bottom part of Table 1: the initialization value `0x55` is read back from `r19` (in red), and because an increment of address `Z` is missing, all the values read back from `r20` to `r25` are shifted (in gray).

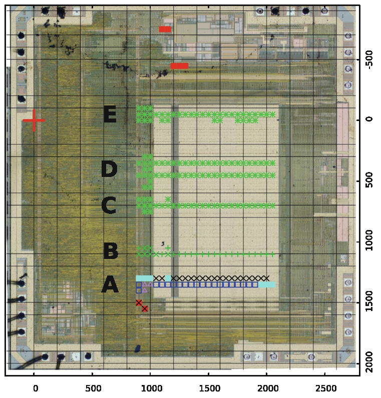
4 Experimental Results

4.1 Finding the Points-of-Interest

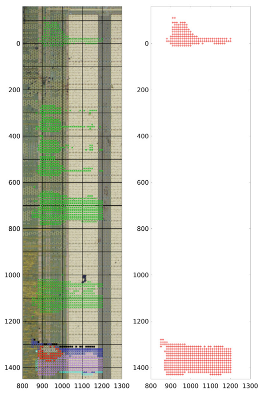
General Overview. Our first series of experiments aimed at finding a point where laser illumination of our target would induce an instruction skip in the test code of Listing 1. Using the synchronization trigger, the delay was set to target the `ld` operation into register `r19`. The laser pulse duration was set to 200 ns (a little more than three clock periods) and its power to 0.5 W. In order to gain a first insight of potential points of interest (i.e. inducing instruction skips), we scanned the whole target area with XY steps of 50 µm. The obtained results are depicted in Fig. 3(a), while the color and shape code describing the faulty behavior is given in Fig. 3(c). The color denotes the number of registers storing the initialization value at read back indicating that the corresponding `ld` operations were not executed (from no register noted S0 to 10 registers noted SA), the shape indicates the number of other incorrect values stored into the registers at read back (from 0 noted E0 to 10 noted EA). As an example, the instruction skip exemplified in the bottom part of Table 1 is depicted by a black × shape (`r19` storing the initialization value and `r20` to `r25` storing six incorrect values, i.e. S1E6).

We identified the following faulty behaviors:

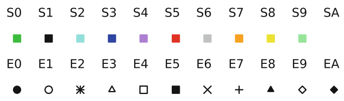
- Red squares on top near an analog block, locations for which the registers read back was all zero. We did not study further this faulty mechanism.
- Red barred squares near the bottom left corner of the Flash memory for which communication with the test chip was lost (until reset).
- Horizontal patterns spanning along the width of the Flash memory (identified with letters from A to E) which were consistent with instruction skips as explained in Table 1.



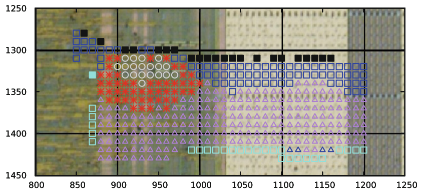
(a) General overview.



(b) Flash left edge.


S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 SA
E0 ● E1 ○ E2 ✳ E3 △ E4 □ E5 ■ E6 × E7 + E8 ▲ E9 ◇ EA ◆

(c) Faults color and shape keys.



(d) Focus on pattern A.

Fig. 3. Laser sensitivity maps obtained while running `Listing 1` test code: general overview (a) and focused overviews (b, d); fault key (c). (Color figure online)

Spatial Focus on Points of Interest. Our next experiments focused on the left edge of the Flash memory with an increased spatial accuracy: XY step set to 10 µm. On its left, Fig. 3(b) reports the observed fault model for a register initialization value of 0x55. The green color of patterns B to E revealed fault models that are not consistent with an instruction skip (the initialization value was not found in any of registers r16 to r25 at read back). We do not report any further analysis of the corresponding fault models since this research work focuses on instruction skips. Note that the authors of [11] obtained similar fault patterns on the same microcontroller, which they identified as bit reset faults induced at read back on the 16-bit opcode instructions.

Instruction Skip Sensitive Area. Fault pattern A reveals a different behavior (see sensitivity map of Fig. 3(d)). Several consecutive instruction skips, from one (depicted in black) to six (depicted in grey), were obtained. The corresponding shapes also revealed that the skips where followed by shifts of the values stored into the registers following the skipped registers. In addition, the right map in Fig. 3(b) shows with red crosses the fault locations where the duration of the trigger signals was shortened by one or more clock periods. It further reinforces our analysis that the instruction skips in pattern A are obtained by turning the `ld` instructions into `nop` instructions. Because execution of a `ld` instruction takes two clock periods contrary to a `nop` instruction which takes one clock period, each consecutive instruction skip shall correspond to a reduction of the test code of one clock period. This phenomenon is displayed in Fig. 4 for a single instruction skip. The test code execution time is shortened as well as the duration of the triggers signals: the fault free execution triggers in blue last one clock period more than the faulted execution that is drawn in red (the third signal, in red, is the laser source `PulseOut` output which shows the actual timing of the laser pulse). For each fault injection location of pattern A, we observed a shortening of the trigger signals equals to the number of instruction skips multiplied by the clock period.

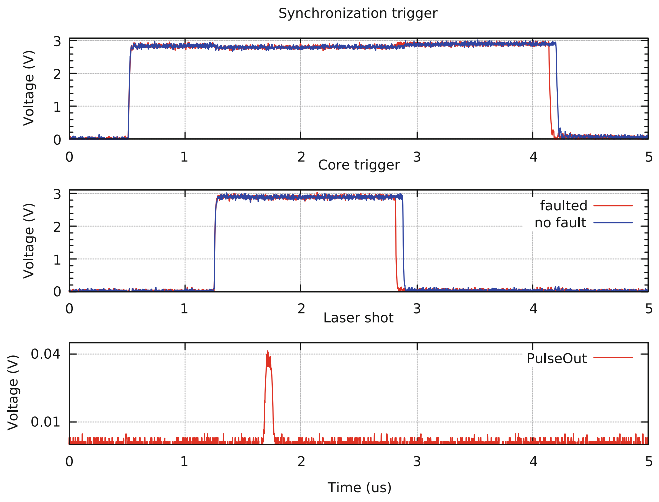
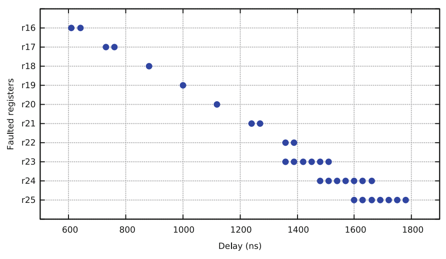


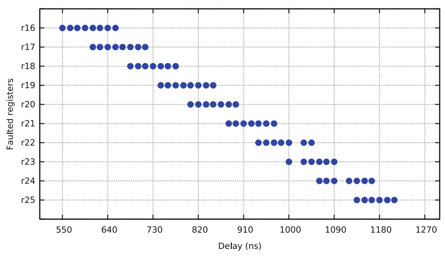
Fig. 4. Synchronisation trigger signals with and without a laser shot (depicted in red and blue resp.), and PulseOut laser source output signal. (Color figure online)

4.2 Laser-Induced Instruction Skip, Test of Accuracy

The left part of pattern A, lying outside the Flash memory, seemed more promising in terms of ability to induce instruction skips. We performed there a set of experiments with an increased XY step accuracy of 5 µm and laser settings ranging from 0.2 W to 0.5 W (0.1 W step) and 50 ns to 125 ns duration (25 ns step). Our objective was to find parameters allowing to induce single instruction skips with high repeatability. We were indeed able to find several locations where we induced such single instruction skips on `r19` with a 100% success rate (the laser parameters were set to 75 ns pulse duration and 0.4 W power). These locations, close to (950 µm; 1,350 µm) (see map of Fig. 3(d)), were used for the experiments reported hereafter.



(a)



(b)

Fig. 5. Faulted registers as a function of laser injection time: laser power of 0.4 W and duration of 75 ns (a) and 125 ns (b).

In terms of accuracy, we also tested whether this single instruction skip fault model was still valid while targeting the `ld` instruction of the other test registers. Our aim was to assess an attacker ability to target arbitrarily a single instruction of a program. To do so, we varied the time delay between the laser shot and the synchronization trigger signal to span the whole test code. Figure 5(a) reports the obtained results. It displays the skipped registers as a function of the delay. It reveals that an attacker is able to inject laser-induced single instruction skips into a running microcontroller with high timing accuracy. For each instruction, we were able to find an injection timing leading to 100% success rate (this success rate may be lower than 100% while passing from one instruction to the other). At some timings, two consecutive instructions were skipped (e.g. at 1,390 ns or 1,625 ns in Fig. 5(a)), suggesting that several consecutive instructions may be skipped simultaneously. For the purpose of verifying this suggestion, we again carried out our experiments with a laser duration increased to 125 ns (two clock cycles) and a delay step set to 20 ns. The corresponding results are displayed in Fig. 5(b). It shows that increasing the laser duration to 125 ns makes it possible to skip two consecutive instructions with a still high timing accuracy (i.e. the ability to choose the two skipped instructions).

4.3 Laser-Induced Arbitrary Number of Instruction Skips

We also tested whether increasing the laser pulse duration would make it possible to skip an arbitrary number of consecutive instructions. The laser power was kept constant at 0.4 W, and the delay was set to target the `ld` instruction of register `r19`. The test series were carried out for a pulse duration ranging from 50 ns to 410 ns with an increment step of 30 ns. Figure 6 reports the obtained results.

A first instruction skip was obtained for a laser pulse duration of 80 ns. Then, the number of instruction skips increased progressively with the pulse duration up to 7 consecutive skips at 350 ns. On average an additional instruction skip was obtained for every 60 ns increment of the laser pulse duration. For each number of instruction skips between 1 and 7, we were able to find settings leading to a 100% success rate.

This suggested that a fault model for which an attacker has the ability to skip an arbitrary number of code instructions (i.e. a chosen number) is feasible. We used the test code shown in Listing 2 to ascertain the highest possible number of consecutive skips. Its structure is similar to the code of Listing 1: a set of target assembly instructions marked by a core trigger encompassed by a larger synchronization trigger with `nop` instructions in between. We chose the `adiw`, or add immediate to word, instruction as target. The target part of the test code had several successive `adiw` instructions used to increment by one the 16-bit word stored in the `r25:r24` registers pair. We chose it over other addition instructions because it lasts two clock cycles, hence each `adiw` faulted into a `nop` shall be ascertained both by the final value stored in `r25:r24` and by a shortening of the trigger signals.

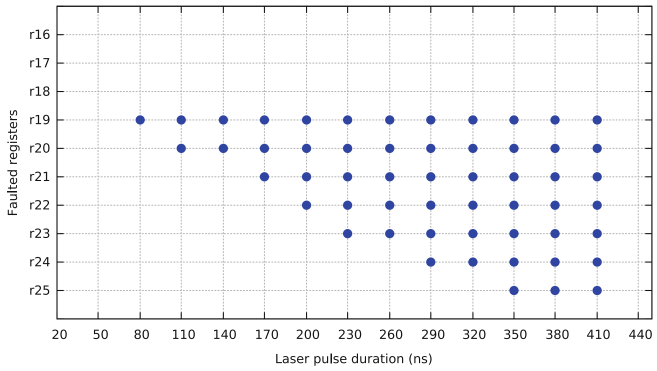


Fig. 6. Faulted registers as a function of the laser pulse duration (laser duration from 50 ns to 410 ns, 0.4 W laser power).

Listing 2 Test code - Max. number of instruction skips.

```
# Initialize r25:r24 at 0x0000
# Set synchronization trigger
nop # 300 ns
# Set core trigger
adiw r24,0x01
...  # 300 times
adiw r24,0x01
# Clear core trigger
nop # 300 ns
# Clear synchronization trigger
# read back r25:r24
```

For our experiments, `r25:r24` was initialized at 0x0000 and the sequence of `adiw` instructions set to 300. The laser power and delay were set to 0.5 W and 800 ns. As we increased progressively the laser pulse duration, an increased number of `adiw` instructions were skipped, for which the shortening of the trigger signals was in accordance with the number of missing additions into `r25:r24`. Table 2 gives the number of obtained successive instruction skips for a selection of laser pulse durations. It took a 20,400 ns long laser pulse to skip the whole 300 `adiw` instructions. Note that the reproducibility of the experiments decreased a bit as the laser pulse duration increased: the number of induced skips varied from 1 or 2 skips at 1,000 ns to 4–5 skips above 10,000 ns from one experiment to the other. We did not test the number of skipped instructions beyond 300. There is a maximal number of instruction skips set by the endurance to laser illumination of the target circuit. Indeed, our device was destroyed when accidentally exposed to a continuous laser pulse at the same 0.5 W power. However, the device we used for these experiments showed no sign of fatigue after several tests at 20,400 ns laser pulse duration.

Table 2. Number of obtained instruction skips vs laser pulse duration

Laser pulse duration (ns)	1,000	2,000	5,000	10,000	20,400
Number of instr. skips	17	33	82	143	300

4.4 PIN Bypass with Several Laser Pulses

The technology of our laser source (see description given in Subsect. 3.1) makes it possible to carve several consecutive pulses in the continuous laser beam delivered by a laser diode. Using such a sequence of laser pulses an attacker might be able to skip several sections of arbitrary length in the target's firmware.

In order to assess the feasibility of this fault injection technique, we targeted a 4-digit PIN verification algorithm (described in [9]). It is protected against side channel timing analysis by a constant-time implementation: every of the digits entered by the user are compared with those of a reference PIN. The four corresponding comparison loops are shown in Listing 3, where `i` is the index of the user and reference PIN arrays (resp. `a1[i]` and `a2[i]`), `PINSIZE` the PIN code length, `BOOL_TRUE` and `BOOL_FALSE` are resp. the true and false boolean values, and `diff` a variable indicating whether the user and reference PINs differ or not. `diff` set to `BOOL_TRUE` indicates that the user and reference PINs are <u>different</u>: as a result the user identification will be rejected (this part of the code is not shown). `diff` is initialized at `BOOL_FALSE` before running the PIN arrays compare loops.

Listing 3 C code of the PIN arrays compare loops.

```
1  BOOL diff = BOOL_FALSE;
2  ...
3  for(i = 0; i < PINSIZE; i++) {
4   if(a1[i] != a2[i]) {
5    diff = BOOL_TRUE;
6   }
7  }
```

Laser-induced fault injection may be used to force the identification of an attacker using a wrong PIN. As illustrated in [8], where a laser-induced modification of one instruction of a similar PIN algorithm permits to perform a PIN bypass. [9] describes, on simulation basis, several instruction skip attacks that may result in a successful PIN bypass: most of them targeting one or a few successive code instructions. We chose to implement that which consists in skipping the four instructions in charge of setting `diff` to `BOOL_TRUE` when a false user PIN is used (line 5 of Listing 3). This fault model is often considered as unlikely because the laser pulse repetition rate of laser sources may be too long. However, our laser is able to meet the 875 ns duration of the PIN compare loops (it is able to emit successive pulses in less than 50 ns). We were able to synchronize four 60 ns long laser pulses in order to skip the instructions used to set `diff` to `BOOL_TRUE` (with a 0.5 W laser power) and gain identification with a false user PIN. Figure 7 displays the core trigger signal encompassing tightly the comparison loops and the `PulseOut` signal showing the actual timing and shape of the four laser pulses.

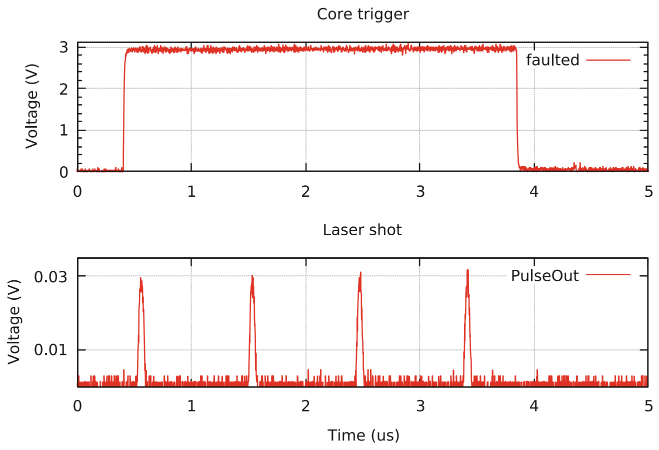


Fig. 7. Laser-induced PIN code bypass induced by four laser pulses emitted in a row (4 × 60 ns, 0.5 W, 875 ns interval).

5 Discussion

5.1 Laser-Induced Instruction Skip Fault Model

A Powerful Fault Model. This work, based on experimental results, studied the instruction skip FM obtained on an 8-bit microcontroller exposed to laser illumination. It demonstrates that a very high accuracy is achievable: we were able to choose and skip a single instruction into a test sequence with a 100% success rate. Moreover, what gives a particular strength to this FM (and where its novelty lie w.r.t. [15,20]) is the ability, by increasing the duration of the laser pulse, to skip an arbitrary number of consecutive instructions of the target's firmware. Provided an attacker is able to find out the adequate laser settings, he shall be able to erase an entire section of a program. Using a laser source with low repetition rate (less than 50 ns in our case), we were also able to skip instructions in the four comparison loops of a PIN code verification algorithm. It made it possible to bypass the PIN verification. This further extends the FM's ability to skip (i.e. erase) several sections of arbitrary length of a microcontroller firmware at runtime.

The strength of this fault model may questions the feasibility of software CMs (a question already raised by [20]): if an attacker has the ability to erase arbitrary sections of a firmware, he will also skip the software CMs it contains. However, secure ICs embed various types of CMs (e.g. laser sensors, hardware redundancy, etc.) that are not put at risk by this FM.

Denomination of the Fault Model. The laser-induced instruction skip FM relies on two possible mechanisms: (1) injecting faults into an instruction opcode [8,11] that turn it into another instruction, or (2) replacing it by a `nop` instruction (this work). In neither case is the instruction really skipped over. The faults we obtained may rather be described as `nop`-ization or as instruction erasure. However, we stuck with the instruction skip denomination for the purpose of avoiding confusion; though, the underlying phenomenon is not an actual skip.

5.2 Synchronization

The fault model we explored is powerful both in terms of accuracy (the ability to skip a single chosen instruction) and of extent (ability to skip several consecutive instructions). However, we used a white box approach for which we used trigger signals to synchronize the laser shots with the test codes. The lack of a trigger signal shall be one of the main difficulty to be tackle with for a real attack case: difficulty of synchronization constitutes a useful and effective counter-measure.

Though, the use of a smart trigger based on real-time pattern recognition of a side channel signal (e.g. the power consumption of the target) may allow an attacker to obtain an accurate synchronization with the operations of its target (as in [18]). Then, the use of a first synchronization may be used to perform a complex attack. As an example, consider the PIN bypass case described in Subsect. 4.4. An attacker may reverse a PIN verification algorithm in a black box scenario for which the PIN algorithm is unknown, provided he owns a valid identification PIN in order to reinitialize periodically the number of PIN trials (it is usually limited to three). This is a four steps process (considering a four digits PIN code). The first step consists in synchronizing a first laser pulse with the first compare loop (see Listing 3) by using a PIN code having a first false digit and three other correct digits. The first synchronization step is achieved when the PIN identification is obtained. Then, iteratively the same process may be used one digit after the other to obtained a full synchronization of the attack. As soon as, the time profile (and laser settings) of a successful PIN bypass is known to the attacker, he will be able to reproduce the attack and gain illegitimate access to whatever is protected by the same PIN verification algorithm.

5.3 Generalization

Our experiments were carried out on a non-secure AVR 8-bit microcontroller which architecture was introduced in 1997. This raises the question of the generalization of the obtained FM to up-to-date microcontrollers. A firm answer to this question is certainly to be based on actual experiments on other targets (e.g. 32-bit microcontrollers).

However, several research works provide indications in favor of generalization. [17] reports laser-induced single instruction skips obtained on a 32-bit cortex-M3 target. The laser sensitive area they identified is located close to the Flash memory of their target, in a similar way of our work. This suggest that a similar mechanism may be at work, even though they did not test the feasibility of successive instruction skips. The authors of [8] also induced single instruction skips through laser illumination into a cortex-M3 microcontroller by faulting a single bit of the targeted instruction (hence inducing an instruction modification equivalent to an instruction skip because the modified instruction had no effect on the executed test code). The physical mechanism they revealed as the root cause of this instruction skip (a laser-induced discharge of a bitline of the Flash memory) appears compatible with induction of several skips if a long laser pulse is used. In terms of accuracy, [10] reports that single bit faults may be induced by laser in targets designed in a technology as advanced as the 28 nm CMOS process. These different research works suggest that a FM for which a single or several successive instruction skips may be feasible with careful tuning of a laser shot parameters and should be considered when designing a secure circuit.

6 Conclusion

This research work assesses on experimental basis an extended fault model for laser-induced instruction skips. The main characteristics of this fault model are:

- its accuracy, or ability to choose the skipped instruction with a 100% success rate provided a precise synchronization is obtained,
- its extension, or ability to skip an arbitrary number of successive instructions,
- its flexibility, or ability to skip several sections of the targeted firmware.

Simply put, laser FA may offer an attacker the ability to erase chosen parts of a microcontroller firmware at runtime. Generalization of this FM beyond the case of our 8-bit target is to be proven, though several results provides arguments in favor of such a possibility. However, this first experimental assessment speaks in favor of considering it when designing CMs. A task that may prove difficult to complete given the assumption that any part of a software CM might be skipped.

Acknowledgment. This research has been partially supported by the European Commission under H2020 SPARTA (Grant Agreement 830892).

References

1. Balasch, J., Gierlichs, B., Verbauwhede, I.: An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs. In: Fault Diagnosis and Tolerance in Cryptography (2011)
2. Barenghi, A., Breveglieri, L., Koren, I., Naccache, D.: Fault injection attacks on cryptographic devices: theory, practice, and countermeasures. Proc. IEEE **100**, 3056–3076 (2012)
3. Beckers, A., et al.: Characterization of EM faults on ATmega328P. In: International Symposium on Electromagnetic Compatibility. IEEE (2019)
4. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_4
5. Breier, J., Jap, D.: Testing feasibility of back-side laser fault injection on a microcontroller. In: Proceedings of the WESS 2015: Workshop on Embedded Systems Security, New York, NY, USA (2015)
6. Breier, J., Jap, D., Chen, C.N.: Laser profiling for the back-side fault attacks: with a practical laser skip instruction attack on AES. In: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, New York, NY, USA (2015)
7. Buchner, S., Miller, F., Pouget, V., McMorrow, D.: Pulsed-laser testing for single-event effects investigations. IEEE Trans. Nuclear Sci. **60**(3), 1852–1875 (2013)
8. Colombier, B., Menu, A., Dutertre, J.M., Moëllic, P.A., Rigaud, J.B., Danger, J.L.: Laser-induced single-bit faults in flash memory: instructions corruption on a 32-bit microcontroller. In: Hardware-Oriented Security and Trust (2019)
9. Dureuil, L., Petiot, G., Potet, M.L., Le, T.H., Crohen, A., de Choudens, P.: FISSC: a fault injection and simulation secure collection. In: International Conference on Computer Safety, Reliability, and Security (2016)
10. Dutertre, J.M., et al.: Laser fault injection at the CMOS 28 nm technology node: an analysis of the fault model, In: 2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (2018)
11. Kumar, D.S.V., Beckers, A., Balasch, J., Gierlichs, B., Verbauwhede, I.: An in-depth and black-box characterization of the effects of laser pulses on ATmega328P. In: Bilgin, B., Fischer, J.-B. (eds.) CARDIS 2018. LNCS, vol. 11389, pp. 156–170. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-15462-2_11
12. Lacruche, M., et al.: Laser fault injection into SRAM cells: picosecond versus nanosecond pulses. In: On-Line Testing Symposium (2015)

13. Moro, N., Heydemann, K., Dehbaoui, A., Robisson, B., Encrenaz, E.: Experimental evaluation of two software countermeasures against fault attacks. In: Hardware-Oriented Security and Trust (2014)
14. Piret, G., Quisquater, J.-J.: A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45238-6_7
15. Rivière, L., Najm, Z., Rauzy, P., Danger, J.L., Bringer, J.: High precision fault attacks on the instruction cache of ARMv7-M architectures. In: 2015 IEEE International Symposium on Hardware Oriented Security and Trust (2015)
16. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_2
17. Trichina, E., Korkikyan, R.: Multi fault laser attacks on protected CRT-RSA. In: Fault Diagnosis and Tolerance in Cryptography (2010)
18. van Woudenberg, J.G.J., Witteman, M.F., Menarini, F.: Practical optical fault injection on secure microcontrollers. In: 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography (2011)
19. Vasselle, A., Thiebeauld, H., Maouhoub, Q., Morisset, A., Ermeneux, S.: Laser-induced fault injection on smartphone bypassing the secure boot. In: 2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (2017)
20. Yuce, B., Ghalaty, N.F., Santapuri, H., Deshpande, C., Patrick, C., Schaumont, P.: Software fault resistance is futile: effective single-glitch attacks. In: 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (2016)

Can Microkernels Mitigate Microarchitectural Attacks?

Gunnar Grimsdal[1], Patrik Lundgren[3], Christian Vestlund[4], Felipe Boeira[2], and Mikael Asplund[2](✉)

[1] Omegapoint, Stockholm, Sweden
gunnar.grimsdal@omegapoint.se
[2] Department of Computer and Information Science, Linköping University, Linköping, Sweden
{felipe.boeira,mikael.asplund}@liu.se
[3] Westermo Network Technologies, Stora Sundby, Sweden
patrik.lundgren@westermo.se
[4] Sectra AB, Linköping, Sweden

Abstract. Microarchitectural attacks such as Meltdown and Spectre have attracted much attention recently. In this paper we study how effective these attacks are on the Genode microkernel framework using three different kernels, Okl4, Nova, and Linux. We try to answer the question whether the strict process separation provided by Genode combined with security-oriented kernels such as Okl4 and Nova can mitigate microarchitectural attacks. We evaluate the attack effectiveness by measuring the throughput of data transfer that violates the security properties of the system. Our results show that the underlying side-channel attack Flush+Reload used in both Meltdown and Spectre, is effective on all investigated platforms. We were also able to achieve high throughput using the Spectre attack, but we were not able to show any effective Meltdown attack on Okl4 or Nova.

Keywords: Genode · Meltdown · Spectre · Flush+Reload · Okl4 · Nova

1 Introduction

It used to be the case that general-purpose operating systems were mostly found in desktop computers and servers. However, as IoT devices are becoming increasingly more sophisticated, they tend more and more to require a powerful operating system such as Linux, since otherwise all basic services must be implemented and maintained by the device developers. At the same time, security has become a prime concern both in IoT and in the cloud domain. This is driven both by increasing regulatory demands as well as end-user expectations in this regard. Putting these two trends together we see that operating system security is now more important than ever.

Last author is partially supported by RICS: the research centre on Resilient Information and Control Systems (www.rics.se) financed by Swedish Civil Contingencies Agency (MSB) and CENIIT project 14.04.

A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 238–253, 2019.
https://doi.org/10.1007/978-3-030-35055-0_15

The *principle of least privilege* is a fundamental pillar in security engineering and dictates that any entity should only have access to the information and resources that it needs to fulfil its purpose. In the context of operating systems, this principle supports the use of *microkernels*, or *microvisors* (i.e., minimal hypervisors operating on the same principle). There are many variants of these, but the basic idea is to have as little of the operating system functionalities implemented in the kernel/hypervisor itself. Services such as process and memory management require the CPU to operate in privileged mode and are therefore part of most microkernels, whereas much of the filesystems and many device drivers can be implemented in user mode (given the right system call interfaces). There are today a number of operating system components and framework developed with security in mind. A prominent example is Genode which is a framework for building secure OSs using a microkernel and provides strong isolation guarantees and resource budgeting for individual components. The basic idea is that Genode enforces a recursive and capability based structure, such that components have exact capabilites and may grant any subset of those capabilites to its children. Genode has been developed to run on multiple kernels, such as Nova, Okl4 and Linux.

The security property of such frameworks which guarantees process isolation hinges on basic assumptions on the underlying hardware that have in recent years been shown not to hold. Attacks such as Meltdown and Spectre and later variants thereof rely on CPU optimisations where the processor performs activities that might be useful in future computations, but which are supposed to be invisible to the processes if they are not used. However, by using some side-channel attack (e.g., involving the cache), the microarchitectural state of these tentative computations can leak to the outside.

It would be naive to assume that a microkernel architecture necessarily protects against microarchitectural attacks. On the other hand, strong isolation properties could potentially mitigate some of the proposed attacks by making some or other step in the attack impossible or less powerful. It has been suggested[1] that the impact should be smaller on Genode than on standard OSs, but so far, there has not been any proper scientific studies on this topic. Schmidt et al. [13] demonstrated ways to circumvent security policies for Genode's IPC and implemented a covert channel which abused a file system cache. However, to the best of our knowledge, there has been no previous work demonstrating a violation of Genode's memory separation.

In this paper we ask the question of whether and if so to what extent a microkernel framework together with state-of-the art secure microkernels such as Okl4 and Nova protects against microarchitectural attacks such as Spectre and Meltdown. Building on previous work (often only provided in blogs and discussion forums) we describe how these attacks can be implemented on three different kernels (Okl4, Nova and Linux), all on top of the Genode framework. Since these attacks are inherently based on time-measurements, we discuss how

[1] N. Feske. *Side-channel attacks(Meltdown, Spectre)*. 2018. URL: https://sourceforge.net/p/genode/mailman/message/36178974/ (visited on 2019-01-16).

to tune the mechanisms to achieve the highest possible throughput, and also what other measures must be taken to make the attacks work. We demonstrate that the underlying side-channel attack (Flush+Reload) of both Meltdown and Spectre works well on all three platforms and that same holds for the Spectre V1 attack. For Meltdown on the other hand, while running without problems on Genode+Linux, we have not been able to show a successful attack on Genode+Okl4 or Genode+Nova. We discuss the reasons for this and potential implications.

The contributions of the paper can be summarised as follows.

– Demonstration of how the Flush+Reload side-channel attack and Spectre V1 attack can be successfully performed on Genode using three different kernels.
– An experimental evaluation of the throughput of Flush+Reload and Spectre achieved under different parameter settings.
– Partial results on the effectiveness of the Meltdown attack.

2 Background and Related Work

In this section, we first give a brief introduction to the two main microarchitectural attacks studied in this paper, Meltdown and Spectre, followed by a description of related work.

2.1 Meltdown and Spectre

Meltdown is a microarchitectural attack which exploits the fact that some modern CPUs may execute instructions out of order [9]. Specifically, Meltdown can read memory from an addressable memory space which it should not be able to read from. Lipp et al. [9] used a Meltdown exploit to read memory from the kernel and other user processes in Linux. This was possible as the Linux kernel's memory was mapped into the address space of each user process. Genode's founder Feske has stated that some in-kernel data structures in Genode are likely vulnerable to the Meltdown attack (see footnote 3).

Spectre relies on the fact that some modern CPUs may speculatively execute instructions [6]. There are different versions of the Spectre attack (e.g., [6,11]), we will be looking at Spectre version 1. Spectre version 1 exploits speculative execution to bypass boundary checks. An attacker could use this attack to execute code which bypasses a boundary check and leaks information to the attacker.

Both Meltdown and Spectre rely on an attacker being able to transmit gathered data to and from the cache. Flush+Reload is a Side-Channel Attack (SCA) which abuses the time difference of fetching uncached and cached data [17]. This channel can be used in the context of Meltdown and Spectre to first read kernel memory into a cache exploiting their respective CPU optimisations. If the address which is cached is carefully crafted, the time with which a process can access this address can be measured to retrieve information.

SCAs extract information from another system or user by abusing some aspects of the system which are not supposed to transmit information. A side channel can also be used as a covert channel, i.e., a channel in which two colluding actors communicate via a side channel.

2.2 Related Work

There has been work on Genode related to security, such as Constable et al. [3] who worked on extending formal Sel4 verification to Virtual Machine Monitor (VMM) running on Genode. Other works have focused on using Genode as a means to achieve a secure OS. Brito et al. [1] used Genode as a secure kernel base to process images securely on an ARM TrustZone cloud environment. However, Genode has seen little work related to microarchitectural attacks and side channels. Schmidt et al. [13] constructed a covert channel in Genode which exploited a software cache to construct a timing channel. However, to the best knowledge of the authors, there has been no other work relating to SCAs in Genode.

Side Channels. Xiao et al. [16] demonstrate a covert channel using execution time for write accesses to shared memory pages. They leverage the Copy-On-Write (COW) technique, which is commonly used for shared memory implementations. They also demonstrate, using this technique, examples of a covert channel transmitting 50–90 bps for practical applications.

Pessl et al. [12] present a covert cross CPU channel utilising varying access times of memory banks in DRAM. They demonstrated a channel with a capacity of 2.1 Mbps with an error probability of 1.8% and across VM channel with a capacity of 596 kbps with an error probability of 0.4%

Microarchitectural Attacks. Mcilroy et al. [11] examined the deep seated implications of how Spectre and incorrect hardware models affect confidentiality-enforcing programming languages. The authors show that these confidentiality guarantees are completely compromised by Spectre. Koruyeh et al. [7] show that the Return Stack Buffer (RSB) could be exploited instead of the BPU, thus introducing a class of SpectreRSB attacks. Koruyeh et al. were not successful in demonstrating these attacks on ARM and AMD CPUs. However, ARM and AMD CPUs also utilise an RSB and should therefore be vulnerable.

There has also been work examining SCAs targeting ARM Trustzone. Lapid and Wool [8] mounted a side-channel cache attack against the ARM32 AES implementation used by the Keymaster trustlet. Another work by Bukasa et al. [2] demonstrate the ineffectiveness of Trustzone to prevent power analysis SCAs.

Microarchitectural attacks are also a quickly progressing field. A recent work by Schwarz et al. demonstrated the ZombieLoad attack, a new type of microarchitectural attack which exploits a fill buffer to read data from other processes [14]. This fill buffer is a type of load queue which is shared between hyper threads. This buffer can under certain circumstances trigger a load which has been initially issued on another core and thereby can leak data from loads issued by other processes [14].

Security by Virtualisation. Using a small kernel is not the only way to potentially enhance the security of a system. Another feasible option is to use different virtual systems to separate processes. The virtual systems need to be running on a hypervisor, which may be attacked. Thongthua and Ngamsuriyaroj [15] discusses some weaknesses they found in popular hypervisor software. However, the abstraction of virtualisation does not prevent microarchitectural attacks such as Meltdown or Spectre [6,9]. Irazoqui et al. [5] recovered an AES key in a cross-virtual machine setup using a SCA that abused the Last-Level Cache (LLC). The attack is not dependent on the virtual machine running on the same core since the LLC cache was used. Virtualisation also adds to overhead by handling multiple OSs running on the hardware.

3 Methodology Overview

In this section we provide an overview of the methodology used in the paper. First, we elaborate on the problem statement by asking three questions regarding the feasibility of performing microarchitectural attacks on the Genode framework. We then proceed to explain our choices of platforms (i.e., what kernels we investigate) and metric (how the attacks have been evaluated).

3.1 Problem Statement

This paper aims to study the impact of microarchitectural attacks on microkernels. In particular, we investigate effectiveness of Meltdown and Spectre on microkernels. Our investigation can be summarised with the following three research questions.

1. Can Flush+Reload be used to create a covert channel between two processes in Genode, measured as the throughput of demonstrated channel?
2. Are Remote Procedure Call (RPC) mechanisms in the microkernels Nova and Okl4 vulnerable to the Spectre Version 1 (Spectre V1) attack, measured as throughput of demonstrated attack?
3. Can the Meltdown attack be executed on Genode?

We try to answer these questions by implementing these attacks on the Genode framework using three different kernels as explained below.

3.2 Choice of Platforms

The overall goal of this paper is to study how well a microkernel architecture can withstand the new class of microarchitectural attacks such as Meltdown and Spectre. There is of course a large number of microkernels available and we have opted to study two of them, Okl4 and Nova. Moreover, we decided to use the Genode framework as a common base for both these kernels as well as in combination with Linux. Genode was chosen since it provides the surrounding services needed to run several different microkernels. Moreover, its strict process separation, adherence to a minimal kernel and open-source code nature make it interesting as a basis for secure operating system design.

We chose two microkernels/microvisors Okl4 and Nova that are designed with security in mind and therefore could potentially provide some protection against the studied attacks. We also tried to use the Sel4 microkernel as it has been formally verified against its specification. Unfortunately, Sel4 on Genode was at the time of our study not well-supported and we did not manage to perform any tests using this kernel.

The Nova kernel, which is a microvisor, is a research project aimed at secure virtualisation. Similar to a microkernel, it provides essential functionality for virtualisation like communication, scheduling and resource management[2].

Okl4 is an open-source microkernel based on the L4 microkernel. It can be used as a hypervisor or as a real-time OS and has been used practically by General Dynamics[3].

3.3 Measuring Attack Effectiveness

To measure the channel's or the attacks' throughput, a fixed string message m of length n was transmitted. Throughput T was then calculated as the number of correctly transmitted bytes per second (Bps) of transmission. This definition of throughput has been used to measure other microarchitectural attacks [7,9]. A byte in position i was considered correctly transmitted if the received byte r_i had the same value as the message byte m_i. The throughput of the channel, T, was calculated as

$$T = \frac{\sum_{i=0}^{n} C(m_i, r_i)}{t_n}, \tag{1}$$

where t_n is the total execution time in seconds, and $C(m, r) = 1$ if $m = r$ and 0 otherwise. An array of size 2048 bytes was used to measure throughput. Every leaked byte was forwarded via serial communication to the measuring system.

Genode's timer object was used in Nova and Linux to measure the total execution time, t_n, with millisecond accuracy. The timer object was not used on Okl4; instead, a timer at the measuring system was used to measure t_n. On Okl4, a start-timer command was transmitted via the serial port before the first transmission byte and an end-timer command after the last byte. The timer on the measuring system was started and stopped by these commands. The execution time, t_n was transmitted after transmitting all bytes if Genode's timer object was used.

4 Attack Implementation

In this section, we first describe how the Flush+Reload channel was implemented, followed by a description of the Meltdown and Spectre implementations.

[2] *NOVA Microhypervisor*. URL: http://hypervisor.org/ (visited on 2019-03-19).
[3] https://gdmissionsystems.com/en/products/secure-mobile/hypervisor.

4.1 Implementing the Flush+Reload Channel

We implemented a Flush+Reload channel on all three platforms. Some adaptations were required such as using the rdtsc instruction rather than rdtscp for time measurements on Nova.

To setup the Flush+Reload channel, we allocated shared memory to a size of (256 + 2) * Padding. There were 256 addresses to distinguish addresses as different values. These addresses were offset using a padding to prevent prefetching between values (another CPU optimisation). Padding was also used at the beginning and at the end of the array to prevent prefetching of shared memory addresses from accesses outside of the array. By performing memory accesses on this array at a given location the memory location is cached and therefore this is indirectly transmitted. The receiver can then measure access times to each address in the array and conclude which corresponding value was transmitted.

Measuring Cache Hits. A threshold was used to decide whether a value was cached or not cached. This threshold was determined by profiling the time it took for the CPU to access cached and uncached values [17]. The Level 1 (L1) cache or LLC was used depending on the attack design. Therefore, two thresholds were defined. One threshold above the L1 cache and one above the LLC.

We assume a memory model of access times as shown in Fig. 1. In this figure, t_{LLC} is the upper bound for the LLC and t_{L1} is the upper bound to access the L1 cache. The thresholds t_{LLC} and t_{L1} are chosen as the upper bound of the measurements for the LLC and L1 cache respectively. This choice was made arbitrarily, with the intent of minimizing false positives while preserving true positives.

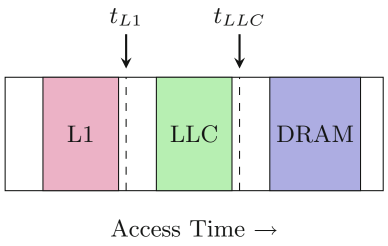


Fig. 1. A model of memory access times for different memory levels.

The time of accessing uncached values was measured by first removing the array from the cache, and then measuring the time for accessing each address. A similar method was used to measure the timings for the L1 cache. Two processes were used to measure the access times to the LLC, one process which cached the values and one process which timed the access time. If the two processes get scheduled on the same core, the values may be cached in either the L1 cache or the LLC.

Preventing Data Prefetching. If a program accesses some sequential data from memory, the CPU will prefetch the coming data items to reduce waiting times. However, since the goal of a Flush+Reload channel is to detect cache hits, this prefetching interferes with these measurements. By adding space between the accessed data items (padding), we can avoid prefecthing. However, too large padding results in excessive memory footprint and slower performance. Therefore, we adopt the use of Strided Read Generator (SRG) [10], which effectively reduces the padding size, while still preventing prefetching. The access pattern is then $x_{si} = ai + b \bmod m$, where $a \equiv 1 \pmod p$ for all prime factors p in m.

The SRGs were evaluated for the padding sizes 4096, 2048, 1024, 512, 256 and 128. The limits 4096 and 128 were used as they are the page size and cache line size on the tested system. Consequently, the CPU does not prefetch for padding sizes over 4096 bytes and padding below 128 bytes does not guarantee separation between values.

All SRGs where $m = 256$, $a \in [1, 255]$ and $b = 0$ were evaluated. The offset $b = 0$ was chosen as a constant offset should not affect prefetching and to limit the number of SRGs to evaluate. Two SRGs are presented, the one with the best performance in Eq. (2) and an arbitrarily chosen worse SRG in Eq. (3). The second is used to illustrate the characteristics of a poor performance SRG.

$$x_i = 49i + 0 \bmod 256 \tag{2}$$

$$x_i = 33i + 0 \bmod 256 \tag{3}$$

Reducing Noise. To obtain a reliable Flush+Reload channel it may be necessary to make multiple measurements, as done by others [7,9]. R different measurements, m_{ij}, were taken for any value i with the purpose of increasing the accuracy. A cache hit detection function f_c was used with a threshold of t_c to build a histogram H of recorded cache hits where each entry h_i is the count of detected cache hits for value i. The estimation $\hat{v}$ of the transmitted value v was calculated as $\hat{v} = \max_i h_i$ where,

$$h_i = \sum_{j=0}^{R} f_c(m_{ij})$$

and,

$$f_c(x) = \begin{cases} 1 & \text{if } x < t_c \\ 0 & \text{otherwise} \end{cases}$$

In addition, synchronising was needed to increase the probability of a successful transmission. Locking was used in order to synchronise the transmitter with the receiver.

4.2 Implementing Meltdown

The methodology for Meltdown was based on the proof-of-concept by Lipp et al. [9]. Specifically, Meltdown required methodologies for recovering from a segmentation fault, identifying a target address, obtaining an observable result via a Flush+Reload channel and synchronising the transmitter with the receiver. On the Linux kernel, we disabled the KPTI patch for the attack to work since the purpose was not to evaluate whether Linux was vulnerable to the attack, but to have it as a baseline implementation.

Recovering from Segmentation Fault. Since Genode does not provide support for segmentation fault handlers [4], another method was needed. One possible method is to start a new child process for each read which leads to a segmentation fault [9]. This method allows for transmitting a single byte with each started child. Another method is to use Intel TSX to suppress the fault [9]. Both methods were evaluated, Intel TSX was chosen due to a more straightforward attack design and fewer resource requirements. If Intel TSX is used, no inter-process synchronisation is needed. A process will continue its execution even if non-accessible memory was accessed during a transaction. The attacker can therefore run Flush+Reload directly after the Meltdown attack.

Choosing a Target Address. Two target addresses were used, the Linux version banner and a victim process. Previous work has had success with these variants[4,5]. Furthermore, they were chosen due to the ease of confirming success using an existing working attack.

In the first alternative, the attacker targets a location for a version string defined in the Linux kernel. Confirmation of correct data was done by reading a file using root privileges.

For the second alternative, a victim process was set up to allocate a secret array of 2048 bytes. The array was cached by the victim. Thereby, the address and value of the target addresses are known, and the addresses along with its values are cached.

4.3 Implementing Spectre

The design of the Spectre V1 attack consisted of an overall design based on previous work[6,7]. Specifically methodologies for ensuring speculative execution, training the branch predictor and increasing accuracy by tuning parameters was used.

The attack setup consisted of a victim process and an attacker which shared a common output buffer. The victim was a vulnerable RPC which accessed an array based on an input index and a bounds check, see Listing 1.1. The attacker exploits this by issuing $T_a - 1$ training requests to a victim_function. After $T_a - 1$ requests the attacker issues a malicious request malicious = target_address with an index targeting an address beyond the bounds of the array. For the attack to work, the vulnerable RPC needs to be speculatively executed and the branch predictor needs to be trained.

[4] https://github.com/paboldin/meltdown-exploit.
[5] https://github.com/IAIK/meltdown.
[6] https://gist.github.com/anonymous/99a72c9c1003f8ae0707b4927ec1bd8a.
[7] https://github.com/crozone/SpectrePoC.

Listing 1.1. Victim Function which is Vulnerable to Spectre V1

```
void victim(size_t idx) {
  if(idx < array_size) {
    int foo = array[idx]; // May speculatively execute
    do_something(foo);    // array_size is not in cache
  }
}
```

Ensuring Speculative Execution. Speculative execution, according to documents from Intel, is highly dependant on microarchitectural implementation and may vary across different processor families. Kocher et al. [6] state that one trigger for speculative execution is a cache-miss prior to or during branch condition evaluation. Therefore, the boundary check values needs to be removed from the cache. This is done with a heuristic flush of the cache by performing a large amount of memory accesses.

Configuring Variables for Spectre. Three parameters are needed to execute the Spectre attack: number of attacks per measurement N_a, the attack period T_a and the number of memory accesses used to flush the cache H_s. Attacks per measurement N_a and T_a were chosen by testing all integers $N_a \in [1, 10]$ and $T_a \in [2, 10]$ to find which combination gave the highest throughput in reading 2048 bytes from the vulnerable process. To determine values for N_a and T_a, H_s was initially chosen to $4096 \cdot 32$, it was then tested using an exponential sample between 64 and the size of the CPU's cache to find a local optimum. It should be noted that the purpose of these local optimisations is not to achieve an optimum, but rather to gauge the possible throughput of this attack.

5 Evaluating Attack Effectiveness

In this section we describe the setup and results of evaluating the effectiveness of the Flush+Reload, Spectre, and Meltdown attacks on the three investigated platforms.

5.1 Setting Up System Under Test

The System Under Test (SUT) is composed of Genode with a microkernel core, an attack implementation and an output channel. This setup was executed on an Intel Core i5-7500 CPU.

We used Genode's build tools and documentation to build our implementation for each kernel[8]. These build tools were available at Genode's Github page[9]. To run a build, Genode requires an init-component which is assigned all system resources. Genode then delegates the task of assigning resources to this

[8] https://genode.org/documentation/developer-resources/index.

[9] https://github.com/genodelabs/genode/tree/18.11.

init-component. We build our implementation by assigning an initial resource budget to our process, thus enabling it to execute, use RPC and allocate memory. Genode's build tools will from our configuration create files which are used to boot the kernel with our implementation. These files can be used by Grub2 to multi-boot the tested SUT.

5.2 Flush+Reload

Since Flush+Reload is a necessary component both in Meltdown and Spectre, and in itself an interesting subject of study in the context of microkernels we show some results both on how to tune this channel for maximum throughput as well as the final performance achieved.

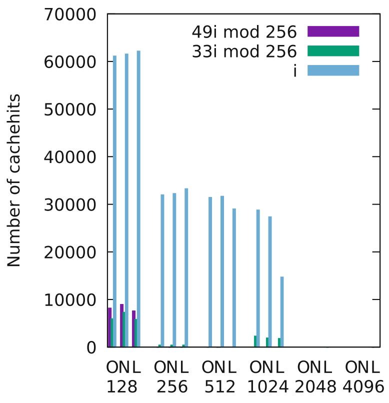
Choosing Cache-Hit Thresholds. Table 1 shows the choices of t_{LLC} and t_{L1} for each kernel along with a valid interval for the choices. The valid interval describes the interval in which there are no measurements from the cache level above and all from the desired one. For example, there are no measurements from LLC below 73 cycles on Okl4. Thus, the valid interval for t_{L1} on Okl4 is $[56, 72]$. The choice of t_{LLC} and t_{L1} was chosen as the lowest value in the valid interval.

Table 1. The Cache-hit thresholds measured in CPU cycles for each kernel.

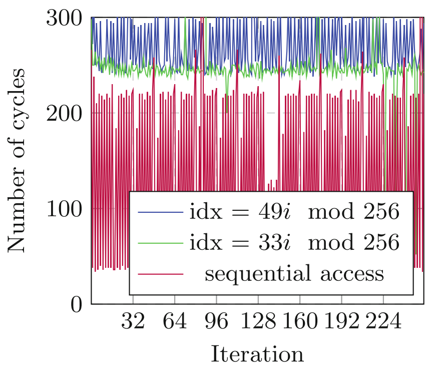
Kernel	Chosen $\mathbf{t_{LLC}}$	Valid interval for $\mathbf{t_{LLC}}$	Chosen $\mathbf{t_{L1}}$	Valid interval for$\mathbf{t_{L1}}$
Okl4	81	$[81, 239]$	56	$[56, 72]$
Nova	80	$[80, 219]$	42	$[42, 64]$
Linux	139	$[139, 239]$	54	$[54, 78]$

Preventing Data Prefetching. Figure 2a shows the number of detected cache hits from the array reads using the SRGs from Eqs. (2) and (3) and different sizes of the internal padding. For each padding size the kernels are denoted using O for Okl4, N for Nova and L for Linux. The SRG $49i \mod 256$ is preventing prefetching at the smallest internal padding and thus results in the smallest memory footprint of the Flush+Reload channel. In Fig. 2b, it can also be seen that the SRG $idx = 49i \mod 256$ results in memory access times of almost 300 CPU cycles which is comparable to DRAM access times. Therefore, the SRG in Eq. (2) and the internal padding of length 256 was used to obtain further results.

Measuring Throughput. Figure 3 shows the throughput of the Flush+Reload channel in six different configurations. For each kernel the transfer was performed within a single process as well as between two different processes. On the x-axis the number of read attempts are shown. Note the logarithmic scale on the x axis.



(a) Number of cache hits from iteration over uncached array using an SRG 256 times on Genode for different internal padding sizes, and different kernels. O = Okl, N = Nova, L = Linux.



(b) Time to access values in a pseudo-randomised or sequential pattern using 256 bytes as internal padding on OKl4.

Fig. 2. Number of cache hits for different padding sizes and kernels (left) and access time for different access patterns (right).

Clearly, the Flush+Reload channel is effective on all three platforms. In four out of the six configurations the throughput is over 1kBps and sometimes much higher than that. However, when transferring data between two different processes on Okl4 and Nova the throughput is less than 100Bps and reduces to just a few Bps for higher number or read attempts. The most likely explanation we could find for this outcome is due to the way the process synchronisation interferes with the data transfer in these setups. It does not seem to be caused by any attack mitigation mechanism in the kernels.

In all six cases, either a single or two read attempts achieves the best throughput, since while repeating the reads reduces the error rate, it also takes longer time.

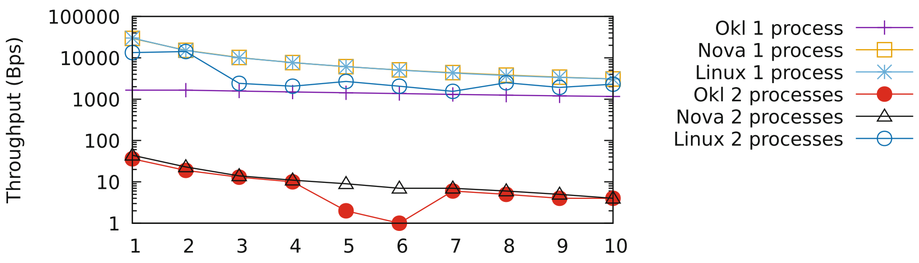


Fig. 3. Reading 2048 bytes within a process or between two processes, for a varying number of attempts

5.3 Meltdown

We were only able to get Meltdown working on the Genode+Linux platform (after disabling the KPTI mitigation). The resulting throughput when read 2048 bytes from another process is shown in Fig. 4. The result shows a fluctuating throughput, ranging from 63 to 11070 Bps. This result demonstrates that the Genode framework in itself does not prevent Meltdown (even if we had to adapt the attack as described in Sect. 4.2)

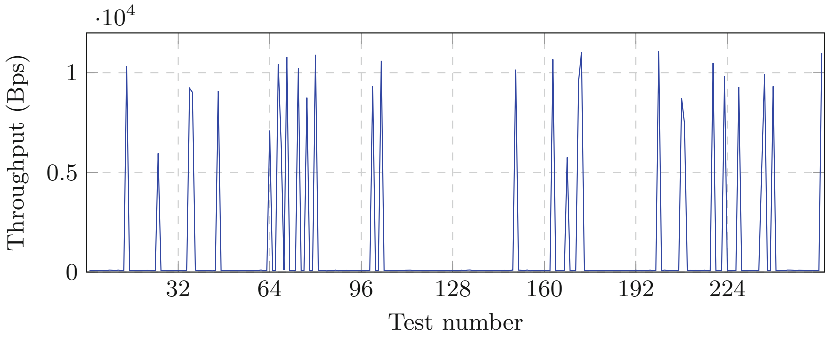


Fig. 4. Throughput from reading 2048 bytes from another process in Genode using Meltdown on Genode+Linux.

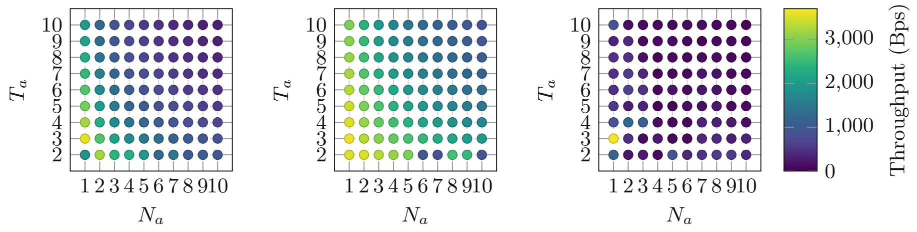
Also interesting is the fact that we were not able to make this attack successful on the Genode+Okl4 or Genode+Nova platforms. Genode makes this attack more difficult to execute by for example not supporting control over which core a process should execute on. However, the main reason we were not able to launch the Meltdown attack on Okl4 or Nova is the lack of mappable address space from another process that can be used in the attack.

5.4 Spectre

We now turn to the Spectre attack. First we show how the parameters were tuned to optimise throughput and then go on to show the resulting throughput.

Attack period, T_a, and number of attacks per measurement, N_a, were tested for $2 \leq T_a \leq 10$ and $1 \leq N_a \leq 10$ on Okl4, Nova and Linux. The result from the tests are shown in Figs. 5a to 5c. The results shows that all the kernels have the highest throughput at $N_a = 1$ and $T_a = 3$. Furthermore, the throughput tends to be lower when N_a or T_a approaches higher values.

We also performed an experiment where H_s was varied to find the value that maximised the throughput. The results indicated that the Spectre V1 attack on Okl4, Nova and Linux had its highest through puts at $H_s = 2^{15}$, 2^{17} and 2^{20} respectively. Note that the difference in H_s varies a factor of 2^5 between kernels, thus, choosing a single value for all kernels is likely not suitable.



(a) Throughput out of for different choices of T_a and N_a when reading a total of 2048 bytes on Genode+Okl4.

(b) Throughput of the Spectre attack for different choices of T_a and N_a when reading a total of 2048 bytes on Genode+Nova.

(c) Throughput of the Spectre attack for different choices of T_a and N_a when reading a total of 2048 bytes on Genode+Linux.

Fig. 5. Throughput of the Spectre attack for different choices of T_a and N_a, when reading a total of 2048 bytes. Then for (a) Throughput for Genode+Okl4, for (b) Throughput for Genode+Nova, and (c) Throughput for Genode+Linux.

The result from trying to read 2048 bytes from an array containing random values with our Spectre V1 implementation is presented in Table 2. The results shows the highest throughput for Nova at 1760 Bps.

Table 2. Result of reading 2048 bytes with Spectre V1 with chosen parameters.

Kernel	Retries	N_a	T_a	H_s	Throughput (Bps)
Okl4	1	1	3	2^{15}	1029
Nova	1	1	3	2^{17}	1760
Linux	2	1	3	2^{20}	525

Clearly, the Spectre V1 attack is effective on all three platforms and with even better performance on the microkernels compared to Linux.

6 Conclusions

In this paper we have examined the vulnerability of microkernels with respect to the microarchitectural attacks Meltdown and Spectre V1. The targeted microkernels were Okl4, Nova and Linux. These kernels were run within the Genode OS framework for evaluation. Relating back to the problem formulation in Sect. 3.1 we draw the following conclusions.

– A covert Flush+Reload channel was demonstrated in Genode with a throughput of 36 Bps on Okl4, 44 Bps on Nova and 13409 Bps on Linux. The large discrepancy between Linux and microkernels deemed likely to stem from scheduling differences.

- The investigated microkernels are vulnerable to Spectre V1 and a proof-of-concept was produced with a throughput 1029 Bps on Okl4, 1760 Bps on Nova and 525 Bps on Linux.
- Results regarding microkernels vulnerability to Meltdown are inconclusive. However, an attack reading the secret of another process in Genode running on Linux was demonstrated with a throughput of 11070 Bps.

Clearly, microkernels and Genode are not secure by design against microarchitectural attacks. Microkernels do have some benefits with regards to mitigating Meltdown as several kernels do not map kernel space into user space and are consequently only affected by Meltdown in a limited way. In addition, Genode does not support for custom segmentation fault handlers. Consequently, the Meltdown attack requires another recovery tool, one such viable option is Intel TSX.

One might ask whether these attacks should be dealt with in software at all or if we should simply wait for chip manufacturers to come up with new chip designs. Intel and AMD have announced fixes to some of the known attacks, but at the same time new ones such as Zombieload and Fallout are being discovered. This does not seem to be a problem that will go away by itself. Moreover, given the huge amount of vulnerable processors already out there, often running critical applications, we cannot just sit back and wait. Perhaps future software systems must fundamentally distrust the hardware on which is running, calling for a completely new security model.

For future work, it would be interesting to find an appropriate target for the Meltdown attack against microkernels in Genode and rigorously attack these targeted addresses. It can also be interesting to pursue another segmentation fault recovery design; this is interesting as Intel TSX is only present on some Intel CPUs. With respect to Spectre V1, it may be interesting to target existing Genode components which expose vulnerable RPCs or implement other Spectre variants which use different techniques, such as variants 2, 3 or SpectreRSB [6,7]. Trying these different variants can further establish the scope of Spectre's impact on microkernels.

References

1. Brito, T., Duarte, N.O., Santos, N.: ARM TrustZone for Secure Image Processing on the Cloud. In: IEEE 35th Symposium on Reliable Distributed Systems Workshops (SRDSW) (2016). https://doi.org/10.1109/SRDSW.2016.17
2. Bukasa, S.K., Lashermes, R., Le Bouder, H., Lanet, J.-L., Legay, A.: How TrustZone could be bypassed: side-channel attacks on a modern system-on-chip. In: Hancke, G.P., Damiani, E. (eds.) WISTP 2017. LNCS, vol. 10741, pp. 93–109. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93524-9_6
3. Constable, S., Sahebolamri, A., Chapin, S.: Extending seL4 Integrity to the Genode OS Framework. Technical report, Critical Technologies (2017)
4. Feske, N.: Foundations: GENODE Operating System Framework 18.05. GENODE LABS (2018). URL: https://genode.org/documentation/genode-foundations-18-05.pdf

5. Irazoqui, G., Eisenbarth, T., Sunar, B.: S$A: a shared cache attack that works across cores and defies VM sandboxing – and its application to AES. In: IEEE Symposium on Security and Privacy (2015). https://doi.org/10.1109/SP.2015.42
6. Kocher, P., et al.: Spectre attacks: exploiting speculative execution. In: 2019 IEEE Symposium on Security and Privacy (SP). IEEE (2019). https://doi.org/10.1109/SP.2019.00002
7. Koruyeh, E.M., Khasawneh, K.N., Song, C., Abu-Ghazaleh, N.: Spectre returns! speculation attacks using the return stack buffer. In: 12th Workshop on Offensive Technologies (WOOT), p. 12 (2018)
8. Lapid, B., Wool, A.: Cache-attacks on the ARM TrustZone implementations of AES-256 and AES-256-GCM via GPU-based analysis. In: Selected Areas in Cryptography (SAC) (2019). https://doi.org/10.1007/978-3-030-10970-7_11
9. Lipp, M., et al.: Melt-down: reading kernel memory from user space. In: 27th USENIX Security Symposium (2018). ISBN: 978-1-939133-04-5
10. Liu, F., Yarom, Y., Ge, Q., Heiser, G., Lee, R.B.: Last-level cache side-channel attacks are practical. In: IEEE Symposium on Security and Privacy. May (2015). https://doi.org/10.1109/SP.2015.43
11. Mcilroy, R., Sevcik, J., Tebbi, T., Titzer, B.L., Verwaest, T.: Spectre is here to stay: an analysis of side-channels and speculative execution (2019). arXiv: 1902.05178
12. Pessl, P., Gruss, D., Maurice, C., Schwarz, M., Mangard, S.: DRAMA: exploiting dram addressing for cross-CPU attacks. In: 25th USENIX Security Symposium (2016). ISBN: 978-1-931971-32-4
13. Schmidt, W., Hanspach, M., Keller, J.: A Case Study on Covert Channel Establishment via Software Caches in High-Assurance Computing Systems (2015). arXiv: 1508.05228 [cs.CR]
14. Schwarz, M., Lipp, M., Moghimi, D., Bulck, J.V., Stecklina, J., Prescher, T., Gruss, D.: ZombieLoad: Cross-Privilege-Boundary Data Sampling (2019). arXiv: 1905.05726 [cs.CR]
15. Thongthua, A., Ngamsuriyaroj, S.: Assessment of hypervisor vulnerabilities. In: International Conference on Cloud Computing Research and Innovations (ICCCRI) (2016). https://doi.org/10.1109/ICCCRI.2016.19
16. Xiao, Y., Zhang, X., Zhang, Y., Teodorescu, R.: One bit flips, one cloud flops: cross-VM row hammer attacks and privilege escalation. In: 25th USENIX Security Symposium (2016). ISBN: 978-1-931971-32-4
17. Yarom, Y., Falkner, K.: Flush+reload: a high resolution, low noise, L3 cache side-channel attack. In: 23rd USENIX Security Symposium (2014). ISBN: 978-1-931971-15-7

MicroSCOPE: Enabling Access Control in Searchable Encryption with the Use of Attribute-Based Encryption and SGX

Antonis Michalas[1], Alexandros Bakas[1](✉), Hai-Van Dang[2], and Alexandr Zalitko[1]

[1] Tampere University of Technology, Tampere, Finland
{antonis.michalas,alexandros.bakas,alexandr.zalitko}@tuni.fi
[2] University of Westminster, London, UK
H.Dang@westminster.ac.uk

Abstract. Secure cloud storage is considered as one of the most important problems that both businesses and end-users take into account before moving their private data to the cloud. Lately, we have seen some interesting approaches that are based either on the promising concept of Symmetric Searchable Encryption (SSE) or on the well-studied field of Attribute-Based Encryption (ABE). Our construction, MicroSCOPE, combines both ABE and SSE to utilize the advantages of each technique. Finally, we enhance our construction with an access control mechanism by utilizing the functionality provided by SGX.

Keywords: Access control · Attribute-Based Encryption · Cloud security · Hybrid encryption · Policies · Storage protection · Symmetric Searchable Encryption

1 Introduction

We are in a period where cloud computing has been established as an essential platform for many businesses looking to build innovative services. However, concerns about security and privacy still remain – especially for companies and users moving their data between multiple *public* cloud services. The main reason behind this, is that most implementations assume an honest and therefore fully trusted cloud service provider (CSP) – a significant obstacle towards enabling a secure cloud posture. Having this in mind, researchers try to address the problem of *secure data storage on untrusted clouds* by looking at how modern cryptographic techniques such as Symmetric Searchable Encryption (SSE) and Attribute-Based Encryption (ABE) can be used to protect users' data.

The original version of this chapter was revised: The spelling of the fourth author's name was corrected. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-35055-0_18

This work was funded by the ASCLEPIOS: Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare Project No. 826093 EU research project.

SSE is a promising encryption technique in which users encrypt their data locally using a symmetric key prior sending them to the CSP. The most exciting thing about SSE though, is that it enables users to search directly over the encrypted data without having the need to decrypt them first. Thus, the CSP learns nothing about the content of the data, except for the information leaked during the execution of the scheme (i.e. access and search pattern). However, a drawback of such scheme is that it does not support efficient revocation since sharing an encrypted file implies sharing the underlying symmetric key.

On the other hand, in ABE schemes all files are encrypted under a master public key but in contrast to traditional public-key encryption, the resulted ciphertext is bounded by a policy. Each user has a unique secret key, which is associated with specific attributes. Hence, decryption of a ciphertext can work if and only if the user's attributes satisfy the policy of the ciphertext. However, using assymetric encryption to encrypt large volumes of data is rather inefficient.

Contribution: In MicroSCOPE we construct a hybrid encryption scheme that utilizes the advantages of both SSE and ABE as well as the functionality offered by SGX [11]. In particular, the symmetric key of the SSE scheme, is encrypted under the ABE scheme and is then stored in an SGX enclave. Moreover, we design an efficient access control mechanism that is agnostic to the ABE scheme.

2 Related Work

In [15], authors use an enclave-based, tree-based search index to design a searchable encryption scheme. Their scheme, HardIDX, addresses the problem of searching in large volumes of encrypted data by making use of the functionalities offered by SGX enclaves. Their solution places the integrity and confidentiality of the search tree in unprotected memory. Thus, every search operation leaks information, such as the access pattern and the size of the output. A promising idea is presented in [14], where the authors present an SGX-based functional encryption scheme called IRON. IRON's main functionalities, such as decryption of a file and application of a function on the decrypted file, both occur in the isolated environment offered by SGX. Moreover, all enclaves can attest to each other and exchange data over secure communication channels. We use the same hardware principles to achieve our hybrid encryption scheme. Another approach is presented in [20] where authors propose a revocable hybrid encryption scheme enhanced with a key-rotation mechanism to prevent key-scrapping attacks. The scheme uses an All-or-Nothing-Transformation (AONT) [10] to prevent revoked users from accessing stored data. More precisely, Optimal Asymmetric Encryption Padding (OAEP) is used due to the fact that reversing OAEP, requires the entire output to be known. As a result, changing random bits of the output renders OAEP's inversion infeasible. However, to decrypt a file, the changed bits need to be stored so that the AONT can be later reversed. To make the scheme more efficient, authors suggest that the AONT could be applied by the server. However, this implies the existence of a fully trusted the server – thus, internal attacks cannot be prevented. In [18] a revocable Ciphertext-Policy ABE

(CP-ABE) scheme in which a revocation list is embedded into the ciphertexts was presented. Naturally, as the revocation list grows, the ciphertexts would become larger, thus rendering any decryption or file modification operation more expensive. To deal with this, a method based on Hierarchical Identity Based Encryption (HIBE) [9], where users' secret keys expire after a certain period of time was proposed. Hence, the revocation list can only include those keys that were revoked before their expiration date. This work is an extension of [7,19], where authors presented a hybrid encryption scheme by combining SSE and ABE. However, their work lacked a proper implementation as well as an access control mechanism like the one we introduce.

3 Cryptographic Primitives

In this section, we present formal definitions for the two main encryption schemes of our construction, CP-ABE and SSE, as described in [8] and [6] respectively.

Definition 1 (Ciphertext-Policy ABE). *A revocable CP-ABE scheme is a tuple of the following five algorithms:*

1. $\mathsf{CPABE.Setup}$ *is a probabilistic algorithm that takes as input a security parameter* λ *and outputs a master public key* MPK *and a master secret key* MSK. *We denote this by* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $\mathsf{CPABE.Gen}$ *is a probabilistic algorithm that takes as input a master secret key, a set of attributes* $\mathcal{A} \in \Omega$ *and the unique identifier of a user and outputs a secret key which is bound both to the corresponding list of attributes and the user. We denote this by* $(\mathsf{sk}_{\mathcal{A},\mathsf{u_i}}) \leftarrow \mathsf{Gen}(\mathsf{MSK}, \mathcal{A}, u_i)$.
3. $\mathsf{CPABE.Enc}$ *is a probabilistic algorithm that takes as input a master public key, a message* m *and a policy* $P \in \mathcal{P}$. *After a proper run, the algorithm outputs a ciphertext* c_P *which is associated to the policy* P. *We denote this by* $\mathsf{c_P} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m, P)$.
4. $\mathsf{CPABE.Dec}$ *is a deterministic algorithm that takes as input a user's secret key and a ciphertext and outputs the original message* m *iff the set of attributes* $\mathcal{A}$ *that are associated with the underlying secret key satisfies the policy* P *that is associated with* c_p. *We denote this by* $\mathsf{Dec}(\mathsf{sk}_{\mathcal{A},\mathsf{u_i}}, c_P) \rightarrow m$.

Definition 2 (Dynamic Index-based SSE). *A dynamic index-based symmetric searchable encryption scheme is a tuple of five polynomial algorithms* $\mathsf{DSSE} = (\mathsf{KeyGen}, \mathsf{InGen}, \mathsf{AddFile}, \mathsf{Search}, \mathsf{Delete})$ *such that:*

- $\mathsf{DSSE.KeyGen}$ *is probabilistic key-generation algorithm that takes as input a security parameter* λ *and outputs a secret key* K. *It is used by the client to generate her secret-key.*
- $\mathsf{DSSE.InGen}$ *is a probabilistic algorithm that takes as input a secret key* K *and a collection of files* $\mathbf{f}$ *and outputs an encrypted index* γ *and a sequence of ciphertexts* $\mathbf{c}$. *It is used by the client to get ciphertexts corresponding to her files as well as an encrypted index which are then sent to the storage server.*

- $\mathsf{DSSE.AddFile}$ *is a probabilistic algorithm that takes as input a secret key* K *and a file* f *and outputs an add token* $\tau_\alpha(f)$ *and a ciphertext* c_f*. The token and the ciphertext are then sent to the storage server, where* c_f *will be added to the collection of ciphertexts and the index* γ *will be updated accordingly.*
- $\mathsf{DSSE.Search}$ *is a deterministic algorithm that takes as input a secret key* K *and a keyword* w *and outputs a search token* $\tau_s(w)$*. The token is then sent to the storage server who will output a sequence of file identifiers* $\mathbf{I}_w \subset \mathbf{c}$*.*
- $\mathsf{DSSE.Delete}$ *is a deterministic algorithm that takes as input a secret key* K *and a file identifier* $id(f)$ *and outputs a delete token* $\tau_d(f)$ *for* f*. The token will be sent to the storage server, who will delete* c_f *and update the index* γ *accordingly.*

To define the security of DSSE we make use of the leakage functions $\mathcal{L}_{in}, \mathcal{L}_s, \mathcal{L}_a, \mathcal{L}_d$ associated to index creation, search, add and delete operations [13].

Definition 3 *(Dynamic CKA 2-security).* *Let* $\mathsf{DSSE} = (\mathsf{KeyGen}, \mathsf{InGen}, \mathsf{AddFile}, \mathsf{Search}, \mathsf{Delete})$ *be a dynamic index based symmetric searchable encryption scheme and* $\mathcal{L}_{in}, \mathcal{L}_s, \mathcal{L}_a, \mathcal{L}_d$ *be leakage functions associated to index creation, search, add and delete operations. We consider the following experiments between an adversary* $\mathcal{ADV}$ *and a challenger* $\mathcal{C}$*:*

Real$_{\mathcal{ADV}}(\lambda)$

$\mathcal{C}$ runs $\mathsf{Gen}(1^\lambda)$ to generate a key K. $\mathcal{ADV}$ outputs a file $\mathbf{f}$ and receives $(\gamma, c) \leftarrow \mathsf{Enc}(\mathsf{K}, f)$ from $\mathcal{C}$. $\mathcal{ADV}$ makes a polynomial time of adaptive queries $q = \{w, f_1, f_2\}$ and for each q he receives back either a search token for w, $\tau_s(w)$, an add token and a ciphertext for f_1, $(\tau_\alpha(f_1), c_1)$ or a delete token for f_2, $\tau_d(f_2)$. Finally, $\mathcal{ADV}$ outputs a bit b.

Ideal$_{\mathcal{ADV},\mathcal{S}}(\lambda)$

$\mathcal{ADV}$ outputs a file $\mathbf{f}$. $\mathcal{S}$ is given $\mathcal{L}_{in}$ and generates (γ, c) which is sent back to $\mathcal{ADV}$. $\mathcal{ADV}$ makes a polynomial time of adaptive queries $q = \{w, f_1, f_2\}$ and for each q, $\mathcal{S}$ is given either $\mathcal{L}_s(\mathbf{f}, w), \mathcal{L}_a(\mathbf{f}, f_1)$ or $\mathcal{L}_d(\mathbf{f}, f_2)$. $\mathcal{S}$ then returns a token and, in the case of addition, a ciphertext c. Finally, $\mathcal{ADV}$ outputs a bit b.

We say that the DSSE *scheme is* $\mathcal{L}$*-i secure if for all probabilistic polynomial adversaries* $\mathcal{ADV}$*, there exists a probabilistic simulator* $\mathcal{S}$ *such that:*

$$|Pr[(Real) = 1] - Pr[(Ideal) = 1]| \leq negl(\lambda)$$

In the cases of file addition and deletion, the simulator must also generate ciphertexts and update the current indexes.

4 Architecture

In this section, we provide an overview of the underlying system model by describing all the different components[1] along with their functionality.

[1] We assume the existence of a registration authority which is responsible for the registration of users. However, registration is out of the scope of this paper and we assume that all users have been already registered.

Cloud Service Provider (CSP): We consider a cloud computing environment similar to the one described in [21]. CSP is responsible for storing encrypted data and must be SGX-enabled since core entities will be running in the trusted execution environment offered by SGX.

Master Authority (MS): MS is responsible for setting up all the necessary public parameters for the proper run of the involved protocols and for generating and distributing ABE keys to the registered users. MS is running in an enclave called the Master Enclave.

Key Tray (KT): KT is a key storage that stores ciphertexts of the symmetric keys generated by various data owners needed to recover data. Every registered user can directly contact KT and request access to the stored ciphertexts. KT is running in an enclave called the KT Enclave.

Revocation Authority (REV): REV is responsible for controlling access rights. REV maintains a mapping of each user with her valid scopes. Each time a scope is revoked from a user, REV updates its database. Similar to MS and KT, REV is also SGX-enabled and is running in an enclave called the REV Enclave.

User (u_i): A user interacts with the CSP to manage certain files that has access to according to her assigned scopes (access rights). The set of access rights of u_i is denoted as $\mathcal{SC}_i = \{(j, s_i^j), \ldots (k, s_i^z)\}$ where $j, \ldots, z$ represent a collection of files encrypted under the symmetric keys $\mathsf{K_j}, \ldots, \mathsf{K_z}$ and s_i^j is a one dimensional bit array of length four that represents the scopes (i.e. view, add, delete, revoke) assigned to u_i for each data collection. For example, if $s_i^j = [1010]$, then u_i has access rights view and delete for data encrypted under the symmetric key $\mathsf{K_j}$.

Threat Model: Our threat model is similar to the one described in [21], based on the Dolev-Yao adversarial model [12]. We extend the above threat model by defining a set of new attacks.

Attack 1 (Successful Scope Substitution Attack – SSA). *Let $\mathcal{ADV}$ be an adversary that corrupts a registered user u_m, whose set of valid scopes is given by s_m^i for data encrypted under the symmetric key $\mathsf{K_i}$. $\mathcal{ADV}$ wishes to tamper with u_m's access rights by providing her with more scopes. We say that $\mathcal{ADV}$ successfully launches an SSA attack iff she can change bits from 0 to 1 in s_m^i and produce a new array $s_m'^i \neq s_m^i$ that will be accepted as valid by the corresponding authorities.*

Attack 2 (Successful Revocation of Legitimate User Attack – RLUA). *Let $\mathcal{ADV}$ be an adversary that corrupts a registered user u_m who has access only to data encrypted under a secret key $\mathsf{K_m}$. Additionally, let u_ℓ be a legitimate user that has access to data encrypted under a secret key K_ℓ, $\ell \neq m$. $\mathcal{ADV}$ successfully launches an RLUA attack iff she manages to revoke scopes to u_ℓ for data that is encrypted under K_ℓ.*

Attack 3 (Successful Compromise of Revoked User Attack – CRUA). *Let $\mathcal{R}_{\mathsf{K_i}}$ be the set of all users that their access to the data encrypted under*

$\mathsf{K_i}$ *has been revoked completely (i.e.* $s^i_{\mathcal{R}_{\mathsf{K_i}}} = [0000]$*). Moreover, let* $\mathcal{ADV}$ *be an adversary that corrupts a user* u_m *where* $u_m \in \mathcal{R}_{\mathsf{K_i}}$*.* $\mathcal{ADV}$ *successfully launches a CRUA attack iff she manages to extract any valuable information about the content of the files that are encrypted with* $\mathsf{K_i}$*.*

5 MicroSCOPE (MSCOPE)

In this section, we present MicroScope (MSCOPE). MSCOPE is built around nine main protocols:

MSCOPE.Setup: Each entity generates a public/private key pair $(\mathsf{pk}, \mathsf{sk})$ for a CCA2 secure public cryptosystem as well as a signing and a verification key for a EUF-CMA secure signature scheme. Furthermore, MS runs CPABE.Setup and generates a master public/private key pair $(\mathsf{MPK}, \mathsf{MSK})$.

MSCOPE.ABEUserKey: This algorithm is executed by a user u_i to receive a secret CP-ABE key. Since MS is responsible for generating such keys, u_i needs to contact MS and request a key. MS will then execute $\mathsf{sk}_{\mathcal{A},\mathsf{u_i}} \leftarrow \mathsf{CPABE.Gen}(MSK, \mathcal{A}, u_i)$, where $\mathcal{A}$ is the set of attributes that is derived from u_i's registered information. Finally, $\mathsf{sk}_{\mathcal{A},\mathsf{u_i}}$ is sent back to u_i over a secure channel.

MSCOPE.Store: After u_i successfully received $\mathsf{sk}_{\mathcal{A},\mathsf{u_i}}$ she can start using the CSP to store files remotely. To do so, she first sends a store request *StoreReq* to the CSP. Specifically, u_i sends $m_1 = \langle r_1, \mathsf{E}_{\mathsf{pk_{CSP}}}(cred_i), StoreReq, \sigma_i(H(r_1||cred_i||StoreReq))\rangle$ where r_1 is a random number. The CSP authenticates u_i as legitimate and sends back an authorization *Auth* as $m_2 = \langle r_2, (Auth), \sigma_{CSP}(H(r_2||u_i||Auth))\rangle$. At this point, u_i generates a symmetric key $\mathsf{K_i}$ to encrypt her files and sends $m_3 = \langle r_3, \mathsf{E}_{\mathsf{pk_{CSP}}}(\mathsf{idx}_{\mathsf{K_i}}), \gamma_i, \mathbf{c_i}, H(r_3||\gamma_i||\mathsf{idx}_{\mathsf{K_i}}||c_i)\rangle$ to the CSP.

MSCOPE.Store

Input: User's authenticator Auth, a collection of files $\mathbf{f_i}$
Output: A collection of ciphertexts $\mathbf{c_i}$ along with an encrypted index γ_i are stored on the CSP

1. u_i sends $m_1 = \langle r_1, \mathsf{E}_{\mathsf{pk_{CSP}}}(cred_i), StoreReq, \sigma_i(H(r_1||cred_i||StoreReq))\rangle$ to the CSP
2. CSP sends : $m_2 = \langle r_2, (Auth), \sigma_{CSP}(H(r_2||u_i||Auth))\rangle$ to u_i
3. u_i runs $(\mathbf{c_i}, \gamma_i) \leftarrow \mathsf{DSSE.Add}(\mathsf{K_i}, \mathbf{f_i})$ and further generates a unique index idx_{K_i} for the symmetric key $\mathsf{K_i}$
4. u_i sends $m_3 = \langle r_3, \mathsf{E}_{\mathsf{pk_{CSP}}}(\mathsf{idx}_{\mathsf{K_i}}), \gamma_i, \mathbf{c_i}, H(r_3||\gamma_i||\mathsf{idx}_{\mathsf{K_i}}||c_i)\rangle$ to the CSP
5. CSP stores $\{\mathbf{c_i}, \gamma_i, \mathsf{idx}_{\mathsf{K_i}}\}$

MSCOPE.KeyTrayStore: Executed by the data owner to store $\mathsf{K_i}$ in KT. First runs $c_P^{\mathsf{K_i}} \leftarrow \mathsf{CPABE.Enc}(\mathsf{MPK}, \mathsf{K_i}, P)$ to obtain $c_P^{\mathsf{K_i}}$ which is then sent to KT via $m_4 = \big\langle \mathsf{r_4}, \mathsf{E}_{\mathsf{pk_{KT}}}(u_i, \mathsf{idx}_{\mathsf{K_i}}), c_P^{\mathsf{K_i}}, \sigma_i\big(H\big(r_4||u_i||c_P^{\mathsf{K_i}}||\mathsf{idx}_{\mathsf{K_i}}\big)\big)\big\rangle$. Then, u_i assigns scopes to the users that wishes to share $\mathbf{c_i}$ with. To do so, she sends

$m_5 = \langle r_5, \mathsf{E}_{\mathsf{pk_{REV}}}(\mathsf{idx}_{\mathsf{K_i}}, \{(u_1, s_1^i), \ldots\}), \sigma_i(H(r_5||\mathsf{idx}_{\mathsf{K_i}}||u_1|| \ldots))\rangle$ to REV. KT stores user's identifier along with the unique index $\mathsf{idx}_{\mathsf{K_i}}$ of the symmetric key, next to $c_p^{\mathsf{K_i}}$.

MSCOPE.KeyTrayStore

Input: $\mathsf{K_i}$, policy P.
Output: KT stores $c_p^{\mathsf{K_i}}$

1. u_i sends $m_4 = \left\langle r_4, \mathsf{E}_{\mathsf{pk_{KT}}}(u_i, \mathsf{idx}_{\mathsf{K_i}}), c_P^{\mathsf{K_i}}, \sigma_i\left(H\left(r_4||u_i||c_P^{\mathsf{K_i}}||\mathsf{idx}_{\mathsf{K_i}}\right)\right)\right\rangle$ to KT and $m_5 = \langle r_5, \mathsf{E}_{\mathsf{pk_{REV}}}(\mathsf{idx}_{\mathsf{K_i}}, \{(u_1, s_1^i), \ldots\}), \sigma_i(H(r_5||\mathsf{idx}_{\mathsf{K_i}}||u_1|| \ldots))\rangle$ to REV
2. KT stores: $\{u_i, c_P^{\mathsf{K_i}}, \mathsf{idx}_{\mathsf{K_i}}\}$
3. REV stores $\langle \mathsf{idx}_{\mathsf{K_i}}, \{(u_1, s_1^i), (u_2, s_2^i), \ldots\}\rangle$ into the list of valid scopes L_{VS}.

MSCOPE.KeyShare: We now assume that another registered user u_j, $j \neq i$ wishes to access $\mathbf{c_i}$. The important thing to notice here is that the data sharing will be done without the involvement of u_i. To this end, u_j sends $m_6 = \langle r_6, \mathsf{E}_{\mathsf{pk_{KT}}}(u_j, u_i), \sigma_j(H(r_6||u_j||u_i))\rangle$ to KT. KT will then reply with $m_7 = \langle r_7, \mathsf{E}_{\mathsf{pk_{REV}}}(u_j, \mathsf{idx}_{\mathsf{K_i}}), \sigma_{KT}(H(r_7||u_j||\mathsf{idx}_{\mathsf{K_i}}))\rangle$. This message will then be forwarded to REV who will locate s_j^i and will send $m_8 = \langle r_8, \mathsf{E}_{\mathsf{pk_{KT}}}(s_j^i)), \sigma_{REV}(H(r_8||s_j^i))\rangle$ to KT. At this point, KT retrieves $c_P^{\mathsf{K_i}}$ and sends $m_9 = \langle r_9, \mathsf{E}_{\mathsf{pk_{CSP}}}(u_j, t, s_j^i, \mathsf{idx}_{\mathsf{K_i}}), c_P^{\mathsf{K_i}}, \sigma_{KT}(H(r_9||u_j||t||s_j^i||\mathsf{idx}_{\mathsf{K_i}}||c_P^{\mathsf{K_i}}))\rangle$ to u_j. Finally, u_j uses her private CP-ABE key to recover $\mathsf{K_i}$.

MSCOPE.KeyShare

Input: User's id u_j and data owner's id u_i
Output: u_j receives $c_p^{\mathsf{K_i}}$

1. u_j sends $m_6 = \langle r_6, \mathsf{E}_{\mathsf{pk_{KT}}}(u_j, u_i), \sigma_j(H(r_6||u_j||u_i))\rangle$ to KT
2. KT replies with $m_7 = \langle r_7, \mathsf{E}_{\mathsf{pk_{REV}}}(u_j, \mathsf{idx}_{\mathsf{K_i}}), \sigma_{KT}(H(r_7||u_j||\mathsf{idx}_{\mathsf{K_i}}))\rangle$ to the user who forwards the message to REV.
3. REV generates and sends $m_8 = \langle r_8, \mathsf{E}_{\mathsf{pk_{KT}}}(s_j^i)), \sigma_{REV}(H(r_8||s_j^i))\rangle$ to KT.
4. KT sends $m_9 = \langle r_9, \mathsf{E}_{\mathsf{pk_{CSP}}}(u_j, t, s_j^i, \mathsf{idx}_{\mathsf{K_i}}), c_P^{\mathsf{K_i}}, \sigma_{KT}(H(r_9||u_j||t||s_j^i||\mathsf{idx}_{\mathsf{K_i}}||c_P^{\mathsf{K_i}}))\rangle$ to the user.

MSCOPE.Search/Update/Delete: Once u_j has gained access to $\mathsf{K_i}$, she can access the encrypted data to either search, add or delete a file (depending on her access rights). To do so, u_j first generates the corresponding DSSE token and sends it to the CSP via $m_{10} = <m_9, token>$. Upon reception, CSP checks if u_j is eligible to perform the operation specified by the token by opening m_9, looking at the timestamp provided by KT and verifying that $s_j^i[n] = 1$, for $n < 3$. If the verifications are correct, CSP proceeds as specified by the DSSE scheme.

MSCOPE.Search/Update/Delete

Input: DSSE token
Output:

1. u_j generates token $= \tau_s(w)/\tau_a(f)/\tau_d(f)$.
2. u_j sends $m_{10} = \langle m_9, \text{token}\rangle$ to the CSP
3. CSP opens m_9 to check if the timestamp is valid and if $s_j^i[n] = 1$, $n < 3$
4. Assuming that all verifications are successful, CSP executes the DSSE algorithm.

MSCOPE.Revoke: A registered user u_j, for whom $s_j^i[3] = 1$, can revoke scopes from other users. To do so, u_j sends $m_{11} = \big\langle r_{11}, \mathsf{E}_{\mathsf{pk}_{\mathsf{REV}}}\big(u_j, u_\ell, n\big), c_P^{\mathsf{K_i}}, \sigma_j(H\big(r_{11}||u_j||u_\ell||n||c_P^{\mathsf{K_i}}))\big\rangle$ to REV, where $n \in [0, 3]$ specifies which scope will be revoked. REV will then send $m_{12} = \big\langle r_{12}, \mathsf{E}_{\mathsf{pk}_{\mathsf{KT}}}\big(u_\ell\big), c_P^{\mathsf{K_i}}, \sigma_{REV}(H\big(r_{12}||c_P^{\mathsf{K_i}}||u_\ell)\big)\big\rangle$ to KT. Upon reception, KT checks if $u_\ell = u_i$ by looking at the value stored next to $c_P^{\mathsf{K_i}}$. If $u_\ell \neq u_i$, KT will send $\mathsf{idx}_{\mathsf{K_i}}$ to REV. REV will retrieve L_{VS} to check whether $s_j^i[3] = 1$ or not. Assuming that the verification is successful, REV removes the specified scope by setting $s_\ell^i[n] = 0$.

MSCOPE.Revoke

Input: u_ℓ, n
Output: $s_\ell^i[n] \to 0$.

1. u_j sends $m_{11} = \big\langle r_{11}, \mathsf{E}_{\mathsf{pk}_{\mathsf{REV}}}\big(u_j, u_\ell, n\big), c_P^{\mathsf{K_i}}, \sigma_j(H\big(r_{11}||u_j||u_\ell||n||c_P^{\mathsf{K_i}}))\big\rangle$ to REV.
2. REV sends $m_{12} = \big\langle r_{12}, \mathsf{E}_{\mathsf{pk}_{\mathsf{KT}}}\big(u_\ell\big), c_P^{\mathsf{K_i}}, \sigma_{REV}(H\big(r_{12}||c_P^{\mathsf{K_i}}||u_\ell)\big)\big\rangle$ to KT.
3. KT verifies that $u_\ell \neq u_i$ and sends $\mathsf{idx}_{\mathsf{K_i}}$ to REV.
4. REV checks if $s_j^i[3] = 1$ and sets $s_\ell^i[n] = 0$.

6 Security Analysis

6.1 Simulation-Based Security

We construct a simulator $\mathcal{S}$ that simulates MSCOPE in a way that any PPT adversary $\mathcal{ADV}$ cannot distinguish between the real protocol and $\mathcal{S}$.

Definition 4 *(Sim-Security). We consider two experiments. In the real experiment, all algorithms run as defined in our construction. In the ideal experiment, a simulator $\mathcal{S}$ intercepts $\mathcal{ADV}$'s queries and replies with simulated responses.*

Real Experiment	*Ideal Experiment*
1. $\mathbf{EXP}^{real}_{MSCOPE}(1^\lambda)$:	1. $\mathbf{EXP}^{ideal}_{MSCOPE}(1^\lambda)$:
2. $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{MSCOPE.Setup}(1^\lambda)$	2. $(\mathsf{MPK}) \leftarrow \mathcal{S}(1^\lambda)$
3. $\mathsf{sk}_{\mathcal{A},\mathsf{u_i}} \leftarrow \mathcal{ADV}^{\mathsf{MSCOPE.ABEUserKey}(\mathsf{MSK},\mathcal{A})}$	3. $\mathsf{sk}_{\mathcal{A},\mathsf{u_i}} \leftarrow \mathcal{ADV}^{\mathcal{S}(1^\lambda)}$
4. $ct \leftarrow \mathsf{CPABE.Enc}(\mathsf{mpk}, m)$	4. $ct \leftarrow \mathcal{S}(1^\lambda, 1^{\lvert m \rvert})$
5. $(\gamma, c) \leftarrow \mathcal{ADV}^{\mathsf{DSSE.Add}(\mathsf{K},\mathbf{f})}$	5. $(\gamma, c) \leftarrow \mathcal{ADV}^{\mathcal{S}(\mathcal{L}_{in}(\mathbf{f}))}$
6. $\mathsf{MSCOPE.Search}(\text{“}search\text{”}, m_s) \rightarrow \mathbf{I_w}$	6. $\mathcal{S}(\text{“}search\text{”}, m_s) \rightarrow \mathbf{I_w}$
7. $\mathsf{MSCOPE.Update}(\text{“}update\text{”}, m_{add}) \rightarrow (\gamma', c')$	7. $\mathcal{S}(\text{“}update\text{”}, m_{add}) \rightarrow (\gamma', c')$
8. $\mathsf{MSCOPE.Delete}(\text{“}delete\text{”}, m_{delete}) \rightarrow (\gamma', c')$	8. $\mathcal{S}(\text{“}delete\text{”}, m_{delete}) \rightarrow (\gamma', c')$
9. *Output b*	9. *Output b′*

We say that MSCOPE is sim-secure if for all PPT adversaries $\mathcal{ADV}$:

$$\mathbf{EXP}^{real}_{MSCOPE}(1^\lambda) \approx \mathbf{EXP}^{ideal}_{MSCOPE}(1^\lambda)$$

At a high-level, $\mathcal{S}$ simulates Key generation and encryption oracles and is given the leakage functions of the DSSE scheme. In the ideal experiment $\mathsf{K_i}$ is not given to $\mathcal{ADV}$. In our game, we exclude MSCOPE.Revoke since L_{VS} is not retrievable during the execution of the protocol. Also, L_{VS} is stored in plaintext and since its values do not depend on sensitive data, side channel attacks on SGX will not reveal any private information.

Theorem 1. *Assuming that* PKE *is an IND-CCA2 secure public key cryptosystem and* Sign *is an EUF-CMA secure signature scheme then MSCOPE is a sim-secure protocol according to Definition 4.*

Proof. We start by defining the algorithms used by the simulator. Then, we will replace them with the real algorithms. Finally, the help of a Hybrid Argument we will prove that the two distributions are indistinguishable.

– MSCOPE.Setup*: Will only generate MPK that will be given to $\mathcal{ADV}$.
– MSCOPE.ABEUserKey*: Will generate a random key to be sent to the adversary. That is, $\mathcal{S}$ simulates CPABE.KeyGen and output $\mathsf{sk}^*_{\mathsf{A},\mathsf{u_i}}$. This key is a random string that has the same length as the output of the real MSCOPE.ABEUserKey*. The key will be given to $\mathcal{ADV}$.
– MSCOPE.KeyShare*: In the ideal experiment, after $\mathcal{ADV}$ requests a secret key, $\mathcal{S}$ will encrypt a sequence of bits under MPK. The ciphertext will be returned to $\mathcal{ADV}$.
– MSCOPE.Search*/Update*/Delete*: When $\mathcal{ADV}$ wishes to generate a DSSE token, $\mathcal{S}$ gets as input the corresponding leakage function $\mathcal{L}$ and outputs a simulated response in accordance with Definition 3.
– MSCOPE.Revoke*: In contrast to the real experiment, the system does not revoke any user.

In a pre-processing phase, the challenger $\mathcal{C}$ generates a symmetric key $\mathsf{K_i}$. We use a hybrid argument to prove that $\mathcal{ADV}$ cannot distinguish between the real and the ideal experiments.

$\boxed{\textbf{Hybrid 0}}$ MSCOPE runs normally.

$\boxed{\textbf{Hybrid 1}}$ Everything runs like in Hybrid 0, but we replace $\mathsf{MSCOPE.Setup}$ with $\mathsf{MSCOPE.Setup}^*$ and $\mathsf{MSCOPE.ABEUserKey}$ with $\mathsf{MSCOPE.ABEUserKey}^*$.

These algorithms are identical from $\mathcal{ADV}$'s perspective and as a result the hybrids are indistinguishable.

After Hybrid 1, we have ensured that $\mathcal{ADV}$ has followed all the required steps in order to ask for $\mathsf{K_i}$. We are now ready to replace $\mathsf{MSCOPE.KeyShare}$ with $\mathsf{MSCOPE.KeyShare}^*$.

$\boxed{\textbf{Hybrid 2}}$ Like Hybrid 1, but $\mathsf{MSCOPE.KeyShare}^*$ runs instead of $\mathsf{MSCOPE.KeyShare}$. Also, the algorithm outputs $\perp$ if $\mathcal{ADV}$ sends m_8 but never contacted REV.

Lemma 1. *Hybrid 2 is indistinguishable from Hybrid 1.*

Proof. By replacing the two algorithms, nothing changes from $\mathcal{ADV}$'s point of view. If $\mathcal{ADV}$ can generate m_8, then she can forge REV's signature. However, this can only happen with negligible probability. So $\mathcal{ADV}$ can only distinguish between Hybrid 3 and Hybrid 2 with negligible probability. □

At this point, $\mathcal{ADV}$ has received what she thinks is a valid $\mathsf{K_i}$. However, $\mathcal{S}$ sent her an encryption of a random string of the same length as $\mathsf{K_i}$. The last part of the proof concerns the DSSE phase of MSCOPE. For the rest of the proof we assume that $\mathcal{ADV}$ performs search, add and delete queries. The simulator now gets access to all leakage functions $\mathcal{L}$ from the DSSE scheme.

$\boxed{\textbf{Hybrid 3}}$ Like Hybrid 2, but when $\mathcal{ADV}$ makes a search, add or delete query, $\mathcal{S}$ is given the corresponding leakage function $\mathcal{L}_i$ and simulates a response. Moreover, the algorithm outputs $\perp$ if $\mathcal{ADV}$ submits such a query but never contacted KT.

Lemma 2. *Hybrid 3 is indistinguishable from Hybrid 2.*

Proof. Assuming the $\mathcal{L}_i-$ security of the DSSE scheme, the token sent by $\mathcal{ADV}$ to the CSP is generated by $\mathcal{S}$ with $\mathcal{L}_i$ as input. As a result, when the CSP receives m_9, it will reply in accordance to Definition 4. Hence, $\mathcal{ADV}$ cannot distinguish between the real and the ideal experiment. Moreover, if $\mathcal{ADV}$ generates m_9 without contacting KT, then she can also forge KT's signature which can only happen with negligible probability. Thus, $\mathcal{ADV}$ can only distinguish between hybrids 4 and 3 with negligible probability. □

With this Hybrid our proof is complete. We managed to replace the expected outputs with simulated responses in a way that $\mathcal{ADV}$ cannot distinguish between the real and the ideal experiment.

SGX Security: Recent works [11,17] have shown that SGX is vulnerable to software attacks. However, according to [14], these attacks can be prevented if the programs running in the enclaves are data-obvious. Thus, leakage can be avoided if the programs do not have memory access patterns or control flow branches that depend on the values of sensitive data. In our construction, no sensitive data (such us decryption keys) are used by the enclaves. KT acts as a storage space for the symmetric keys and does not perform any computation on them. Hence, all the $c_p^{\mathsf{K_i}}$ are data-obvious. Moreover, L_{VS} is stored in plaintext and every entry in the list is padded to achieve same length. Finally, we can prevent timing attacks on L_{VS} by ensuring that every time REV accesses the list either sends back a token, or add a new id, it goes through the entire list.

6.2 Protocol Security

Proposition 1 (SSA Soundness). *Let $\mathcal{ADV}$ be a malicious adversary that corrupts a user u_m with valid scopes s_m^j for data encrypted under a symmetric key $\mathsf{K_j}$. Then $\mathcal{ADV}$ can not successfully perform an SSA attack.*

Proof. Assume that $\mathcal{ADV}$ produces a new set of valid scopes $\mathcal{SC}'_m$ and successfully replaces u_m's valid scopes $\mathcal{SC}_m$. To do so, $\mathcal{ADV}$ needs to successfully flip at least one bit $s_m^j[n]$ to its opposite value $\overline{s_m^j[n]}$, resulting to the new scope array $s_m^{'j}$. We examine ADV's behavior by analyzing two distinct cases:

α. If $\mathbf{0 \le n \le 2}$, then $s_m^j[n]$ will correspond to one of the following: $\{\mathsf{view}, \mathsf{add}, \mathsf{delete}\}$. In that case, $\mathcal{ADV}$ sends $m = <m_8, \text{token}>$, where token is the component of the message which is associated with search, add or delete operations. However, $\mathcal{ADV}$ cannot know $\mathsf{idx}_{\mathsf{K_i}}$ and thus, can never construct this message. As a result $n \notin [0, 2]$

β. If $\mathbf{n = 3}$, then $s_m^j[n]$ corresponds to the scope revoke. $\mathcal{ADV}$ sends $m_{11} = \langle r_{11}, \mathsf{E}_{\mathsf{pk_{REV}}}(u_j, u_\ell, n), c_P^{\mathsf{K_i}}, \sigma_j(H(r_{11}||u_j||u_\ell||n||c_P^{\mathsf{K_i}}))\rangle$ (n_ℓ is the index of the one dimensional array s_ℓ^j specifying which scope to be revoked for u_ℓ) to REV. At this point, REV will retrieve the list of scopes L_{VS} and will check whether $s_m^j[3] = 1$ or not. Since $\mathcal{ADV}$ can not tamper with L_{VS}, REV will see that $s_m^j[3] = 0$ and the attack will fail.

As a result, the attack will fail $\forall n \in [0, 3]$ and this concludes our proof. □

Proposition 2 (RLUA Soundness). *Let $\mathcal{ADV}$ be a malicious adversary that corrupts a user u_m with access rights $\mathcal{SC}_m$. Furthermore, let u_ℓ be a legitimate user, with access rights $\mathcal{SC}_\ell \neq \mathcal{SC}_m$. Moreover, we assume that $\exists j : (j, s_\ell^j) \in \mathcal{SC}_l$, and $(j, s_m^j) \notin \mathcal{SC}_m$. Then $\mathcal{ADV}$ cannot successfully perform an RLUA attack.*

Proof. User u_m launches an attack to u_ℓ by sending $m_{11} = \langle r_{11}, \mathsf{E}_{\mathsf{pk_{REV}}}(u_m, u_\ell, n), c_p^{\mathsf{K_j}}, \sigma_m(H(r_{11}||u_m||u_\ell||n||c_p^{\mathsf{K_j}}))\rangle$ to REV. Upon reception, REV checks the integrity and the freshness of the message. Since this message can be constructed by anyone, REV proceeds by contacting KT to receive $\mathsf{idx}_{\mathsf{K_j}}$. Upon reception of the index, REV retrieves the list L_{VS} to check whether $\langle \mathsf{idx}_{\mathsf{K_j}}, \{u_m, s_m^j[3] = 1\}\rangle \in L_{VS}$ or not. However, since $(j, s_m^j) \notin \mathcal{SC}_m$, $s_m^j[3] \neq 1$ the verification fails and $\mathcal{ADV}$ cannot successfully launch the attack. □

Proposition 3 (CRUA Soundness). *Let $\mathcal{ADV}$ be a malicious adversary that corrupts a user u_m whose access to data encrypted under $\mathsf{K_i}$ has been revoked (i.e. $u_m \in \mathcal{R}_{\mathsf{K_i}}$). Then $\mathcal{ADV}$ cannot successfully perform a CRUA attack.*

Proof. $\mathcal{ADV}$ can successfully perform such an attack if and only if both of the following conditions hold:

α. **$\mathcal{ADV}$ can access the symmetric key $\mathsf{K_i}$.**
Since $u_m \in \mathcal{R}_{\mathsf{K_i}}$, this is always true.

β. **$\mathcal{ADV}$ can bypass the authentication of the different components of the system model.**
For condition β to hold, CSP must be convinced that $u_m \notin \mathcal{R}_{\mathsf{K_i}}$. To do so, $\mathcal{ADV}$ must generate a valid m_8 which can be done with the following ways:

– **Replay an old message:** $\mathcal{ADV}$ sends m_9 she received from KT during the execution of MSCOPE.KeyShare. Since m_9 used to be valid, $\exists\ n : s^i_m[n] \neq 0$ and it also contains a valid signature from KT. However, since the timestamp contained in this message is not fresh, verification will fail.

– **Impersonate a legitimate user:** Another approach for u_m would be to impersonate a registered user u_ℓ, such that $s^i_\ell \neq [0000]$. We assume that u_m obtains $m_9 = \langle r_9, \mathsf{E}_{\mathsf{pk_{CSP}}}(u_j, t, s^i_j, \mathsf{idx_{K_i}}), c_P^{\mathsf{K_i}}, \sigma_{KT}(H(r_9||u_j||t||s^i_j|| \mathsf{idx_{K_i}}||c_P^{\mathsf{K_i}})\rangle$ to u_j and tries to tamper with it. However, without knowing the index of the encryption key $\mathsf{idx_{K_i}}$, she cannot alter the first part of the message and replaces u_ℓ with u_m. Therefore, the attack will again fail.

Hence, only one of the two conditions holds. Therefore, the attack fails. □

Table 1. Size of datasets and keywords

TXT files	Dataset size	Unique keywords
425	184 MB	1,370,023
815	357 MB	1,999,520
1,694	670 MB	2,688,552
1,883	1 GB	7,453,612
2,808	1.7 GB	12,124,904

Table 2. Keywords and filenames pairs

Unique keywords	(w, id) pairs
1,370,023	5,387,216
1,999,520	10,036,252
2,688,552	19,258,625
7,453,612	28,781,567
12,124,904	39,747,904

7 Experimental Results

For the implementation of the CP-ABE scheme, we used the library provided by Bethencourt et al. [8], while for the DSSE scheme we used the one described in [6] and for the parts that run in enclaves we used the SGX-OpenSSL library [3].

To evaluate the performance of MicroSCOPE under realistic conditions, we used different machines – depending on the process to be measured. The setup of the DSSE scheme was measured on a Microsoft Surface Book laptop with a 2.1 GHz Intel Core i7 processor and 16 GB RAM running Windows 10 64-bit.

The reason being that in a practical scenario, this process would take place on a user's machine. Conducting the experiments on a powerful server would result in a set of non-realistic results. The parts running in an enclave were measured in a powerful desktop PC with Intel Core i7-8700 at 3.20 GHz (6 cores), 32 GB of RAM running Ubuntu 64-bit and Intel SGX Hardware Debug mode build configurations. The reason for running these parts on such a computer is based on the assumption that these processes will be running on the CSP.

7.1 Symmetric Searchable Encryption

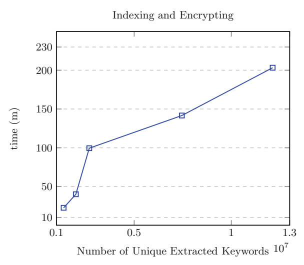
This part of the experiments was implemented in Python 2.7 using the PyCrypto [1] library. To test the overall performance of the DSSE scheme, we used files of different size and structure from the Gutenberg dataset [2]. Our experiments focused on two main aspects: *(1)* Indexing and *(2)* Searching for a specific keyword. Additionally, our dictionaries were stored in a MySQL database.

Dataset: We created five different datasets with random text files from the Gutenberg dataset. The selected datasets ranged from text files with a total size of 184 MB to a set of text files with a total size of 1.7 GB. Using pure text files resulted in a very large number of extracted keywords (12 million distinct keywords without counting the stop words). Table 1 shows the different datasets we used along with the total number of the extracted unique keywords.

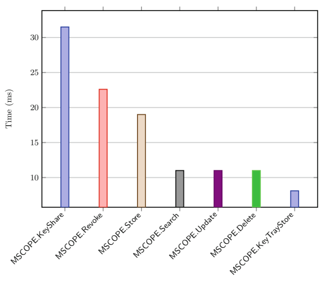
Indexing & Encryption: Indexing is the setup phase of the DSSE scheme during which the following steps take place: *(1)* reading plaintext files and generating the dictionary, *(2)* encrypting files, and *(3)* building the encrypted indexes. We measured the total setup time for the datasets shown in Table 1. We ran each process ten times and measured the average completion time. Figure 1a illustrates the time needed for indexing and encrypting text files ranging from 184 MB to 1.7 GB. To index and encrypt text files that contained 1,370,023 distinct keywords the average processing time was 22.48 min while for 12,124,904 distinct keywords the average processing time was 203.28 min. Considering that this phase is the most demanding one, the required time is considered as acceptable not only based on the size of the datasets but also based on the results of other schemes that do not offer forward privacy [13] as well as on the fact that we ran our experiments on a commodity laptop.

Apart from the unique keywords indexer, the incorporated DSSE scheme also creates an indexer that maintains a mapping between a keyword (w) and the filename (id) that w is contained. The total number of the generated pairs in relation to the size of the underlying datasets is shown in Table 2.

Search: On average, the time needed to generate the search token is 9 μs while the actual matching of the files that contain the keyword that is being searched is just a `SELECT` and `UPDATE` query to the database. Searching for a specific keyword over a set of 12,124,904 distinct keywords and 39,747,904 addresses required 3.2 s.



(a)



(b)

Fig. 1. (a) Indexing and encrypting files (b) message creation and verification

7.2 Implementation/Evaluation of MicroSCOPE

We used the SGX OpenSSL cryptographic library [3] to implement the RSA cryptosystem with 4096-bit key sizes, and a set of cryptographic hash functions. Development was done in C using Intel(R) SGX SDK 2.6 for Linux [4].

In an SGX environment there are two main components: *(1)* the trusted component (enclave), and *(2)* the untrusted component (application). The untrusted application makes a call to an enclave by using SGX's `ECall` function which allows the application to enter the enclave. To temporarily exit the enclave and call a function in untrusted space, SGX's function `OCall` is used.

Enclave Creation & Key Generation: First, we measured the time needed to launch the four enclaves (MS, CSP, KT, and REV). Each enclave contains a different set of functions that corresponds to different parts of the protocol. We launched each enclave 10,000 times and measured the average completion time. The time to launch the MS enclave, containing functions for the generation of fresh RSA key pairs, was 25.29 ms while the time to launch the REV enclave, containing MSCOPE.KeyShare and MSCOPE.Revoke functions, was 27.19 ms. Time required to launch the KT enclave, containing MSCOPE.KeyShare, MSCOPE.Revoke and MSCOPE.KeyTrayStore functions, was 28.3 ms while for the CSP enclave that contained MSCOPE.Store, MSCOPE.Search, MSCOPE.Update, and MSCOPE.Delete functions, 28.12 ms was required. Since the process of launching enclave can run in parallel, the average time required by MicroSCOPE to launch all four enclaves is 28.3 ms. Table 3 summarizes the results by presenting the time needed to launch each enclave of MicroSCOPE as well as the specific functions from the protocol that each enclave contains. Additionally, we also measured the time needed to launch an empty enclave. Launching an empty enclave took on average 9.2 ms. Even though launching the enclaves of MicroSCOPE needed 28.3 ms the difference of 19 ms is considered as negligible considering that the setup of MicroSCOPE runs only once.

Each enclave generates an RSA key pair of 4096-bit length. As can be seen in Table 3 the average time to generate an RSA key pair of 4096-bit size was

840 ms. Again, this is a process that can run in parallel. As a result, the total time required for a complete setup of the enclaves is estimated at 921.71 ms.

Enclave Attestation: Intel's SGX supports two forms of attestation: Local attestation, and Remote attestation. Local attestation involves two or more enclaves running on the same platform, whereas the Remote attestation allows a remote third party to attest an enclave. This currently demands to contact Intel's Attestation Server – a process that requires a license. As a result, in our experiments we only measured the time needed to perform Local Attestation between REV and KT enclaves. We ran local attestation 10,000 times and found that the average time needed to successfully complete the process was 1.1 ms.

Execution Time: To evaluate the total execution time we measured the running time of MicroSCOPE's core functions by calculating the time to generate, send and verify the exchanged messages. We ran each function 100,000 times and calculated average execution time. Our primary focus was to measure the execution time of all involved `ECalls` and `OCalls`. Figure 1b summarizes the results of this experiment by showing the average processing time needed for each one of the core functions. As can be seen in Fig. 1b, MSCOPE.KeyShare and MSCOPE.Revoke are the two most demanding processes. MSCOPE.KeyShare needed on average 31.5 ms while 22.6 ms was the execution time of MSCOPE.Revoke. However, the running time of MSCOPE.Revoke also depends on the length of the revocation list. Currently, we only measured the time to construct, send and verify the required messages. Moreover, the execution time for MSCOPE.Store was measured at 19 ms. The corresponding times needed for MSCOPE.Search, MSCOPE.Update and MSCOPE.Delete appeared to be the same – 11 ms on average. Finally, MSCOPE.KeyTrayStore appeared to be the lightest function with an average execution time of 8.1 ms.

Table 3. Setup time for main MSCOPE components

Enclave creation	MSCOPE functions	Time
RSA setup	Average time for generating a 4,096 bit long RSA key pair	840 ms
EMPTY enclave	Average time for launching an empty enclave	9.2 ms
MS encalve	Containing MSCOPE's key generation functions	25.29 ms
REV enclave	MSCOPE.KeyShare	27.18 ms
KT enclave	MSCOPE.KeyShare MSCOPE.KeyTrayStore MSCOPE.Revoke	28.3 ms
CSP enclave	MSCOPE.Store MSCOPE.Search MSCOPE.Update MSCOPE.Delete	28.12 ms
Local attestation	KT & REV	1.1 ms

7.3 Ciphertext-Policy Attribute-Based Encryption

MSCOPE only uses CP-ABE to encrypt a symmetric key and *not* large volumes of data. To this end, we measured the time needed to encrypt and decrypt a symmetric key under policies of different sizes. We used access policies of the type '`Attribute_1 AND ...AND Attribute_n`' as in [5]. Such policies are the most demanding since all attributes are required for the decryption. For the encryption of a file with a policy consisting of the conjunction of 1000 attributes the time needed was 11 s, while the corresponding decryption time was measured at 4.5 s. However, for a realistic scenario in which the policy consists of 200 attributes, the encryption and decryption times were measured at 2.1 s and 0.6 s respectively. Thus, the use of CP-ABE scheme does not put any real computational burden to the performance of MSCOPE.

8 Conclusion

We proposed a hybrid encryption scheme that combines SSE and ABE in a way that the advantages of each encryption technique are used. MicroSCOPE allows clients to search over encrypted data while the symmetric key required for the decryption is protected via a CP-ABE scheme. Our construction allows data owners to share their data based on certain access rights. Finally, we have shown how to rely on SGX to provide an efficient revocation mechanism that is agnostic to the underlying encryption schemes.

References

1. PyCrypto - the Python cryptography toolkit. pypi.org/project/pycrypto/
2. Project Gutenberg (1971). https://www.gutenberg.org/
3. SGX-OpenSSL (2017). https://github.com/sparkly9399/SGX-OpenSSL
4. SGX-SDK (2019). https://01.org/intel-softwareguard-extensions/downloads/intel-sgx-linux-2.6-release
5. Agrawal, S., Chase, M.: FAME: fast attribute-based message encryption. In: Proceedings of CCS 2017. ACM (2017)
6. Bakas, A., Michalas, A.: Multi-client symmetric searchable encryption with forward privacy. Cryptology Archive, 2019/813. https://eprint.iacr.org/2019/813
7. Bakas, A., Michalas, A.: Modern family: a revocable hybrid encryption scheme based on attribute-based encryption, symmetric searchable encryption and SGX. Cryptology ePrint Archive, Report 2019/682 (2019). https://eprint.iacr.org/2019/682
8. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy. IEEE (2007)
9. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_26
10. Boyko, V.: On the security properties of OAEP as an all-or-nothing transform. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 503–518. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_32

11. Costan, V., Devadas, S.: Intel SGX explained (2016)
12. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)
13. Dowsley, R., Michalas, A., Nagel, M., Paladi, N.: A survey on design and implementation of protected searchable data in the cloud. Comput. Sci. Rev. **26**, 17–30 (2017)
14. Fisch, B., Vinayagamurthy, D., Boneh, D., Gorbunov, S.: Iron: functional encryption using Intel SGX. In: Proceedings of CCS 2017. ACM (2017)
15. Fuhry, B., Bahmani, R., Brasser, F., Hahn, F., Kerschbaum, F., Sadeghi, A.-R.: HardIDX: practical and secure index with SGX. In: Livraga, G., Zhu, S. (eds.) DBSec 2017. LNCS, vol. 10359, pp. 386–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61176-1_22
16. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption, pp. 965–976 (2012)
17. Lee, S., Shih, M.W., Gera, P., Kim, T., Kim, H., Peinado, M.: Inferring fine-grained control flow inside SGX enclaves with branch shadowing. In: 26th USENIX Security Symposium (2017)
18. Liu, J.K., Yuen, T.H., Zhang, P., Liang, K.: Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list. In: Preneel, B., Vercauteren, F. (eds.) ACNS 2018. LNCS, vol. 10892, pp. 516–534. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93387-0_27
19. Michalas, A.: The lord of the shares: combining attribute-based encryption and searchable encryption for flexible data sharing. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019. ACM (2019)
20. Myers, S., Shull, A.: Practical revocation and key rotation. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 157–178. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76953-0_9
21. Paladi, N., Gehrmann, C., Michalas, A.: Providing user security guarantees in public infrastructure clouds. IEEE Trans. Cloud Comput. **5**(3), 405–419 (2017). https://doi.org/10.1109/TCC.2016.2525991

VMIGuard: Detecting and Preventing Service Integrity Violations by Malicious Insiders Using Virtual Machine Introspection

Stewart Sentanoe(✉), Benjamin Taubmann, and Hans P. Reiser

University of Passau, Passau, Germany
{se,bt,hr}@sec.uni-passau.de

Abstract. Organizations often focus their IT security strategy on protecting the perimeter from outside attacks, but internal attacks can often cause the greatest damage. Version control systems are frequently used in software development, including processes for automated build and deployment. Malicious insider manipulations in a version control system can, for example, lead to a clandestine distribution of software with implanted vulnerabilities, backdoors, or other malicious functionality.

In this paper, we present VMIGuard, a solution that leverages virtual machine introspection (VMI) to detect integrity violations and prevent the propagation of unauthorized changes to a version control system caused by an insider attack. VMIGuard logs metadata about all authenticated modifications, and for each retrieval of version control system content, it verifies on-the-fly if the retrieved content matches the expected state. VMIGuard prevents the delivery of manipulated version control system content and notifies the user about integrity violations. We evaluate VMIGuard based on the open-source version control system git with several scenarios, in which it increases the response time in the worst case of the version control system server by a maximum of only 10%.

Keywords: Integrity protection · Virtual machine introspection · Version control systems · Malicious insider

1 Introduction

Over the past two decades, there has been a dramatic increase in the need for reliable services that allow software developers to collaborate from all over the world and merge their code easily. Today, developers rely on a version control system (VCS) to ensure their collaboration going smoothly.

In recent years, git [14] has evolved to be the most widely used VCS. Developers can use git via third-party hosted services such as GitHub [6], GitLab [7] and BitBucket [3].

Developers can also self-host a VCS by installing VCS solutions such as Gogs [8], Gitea [5], or Gitlab [7] on their infrastructure. The major advantages

A. Askarov et al. (Eds.): NordSec 2019, LNCS 11875, pp. 271–282, 2019.
https://doi.org/10.1007/978-3-030-35055-0_17

of self-hosted solutions are that the developers have full control of their data and such a VCS can be placed inside a private network that is not accessible publicly.

Developers rely on the VCS services to accurately store their code and the history of their modification. Besides, VCS systems are frequently used to ensure smooth workflows. For example, a VCS can allow modifications to a master or release branch only after review and approval. A VCS such as git provides reasonable mechanisms to enforce such guarantees if all changes are made via a (web) service that enforces access control. However, insider attacks, such as malicious administrators who have low-level access to the server, can bypass access control and make unauthorized changes that compromise the integrity of the repository.

To mitigate that problem, we present VMIGuard that helps users to detect integrity violations caused by a malicious insider. The goals of VMIGuard are the following: VMIGuard should be able to detect manipulations of a git repository, and prevent their propagation to clients. We do not want to design a completely new VCS or substantially modify an existing one. Instead, we aim at designing a generic security solution that transparently enhances an existing VCS with the desired properties. As a basis for our work, we choose git, because today it is the most commonly used VCS.

VMIGuard uses virtual machine introspection (VMI), which is a technique to analyze the state (passive VMI) and the control flow (active VMI) of a virtual machine (VM) from the hypervisor point of view [9]. VMI allows us to access all information regarding a VM state such as CPU registers and memory. For example, VMI allows us to access the decrypted data from encrypted network data in memory. We demonstrate that VMIGuard can offer the desired additional protection. We measure the performance of VMIGuard with some scenarios where it increases the response time by at most 10% in the worst case and 6.5% in the best case.

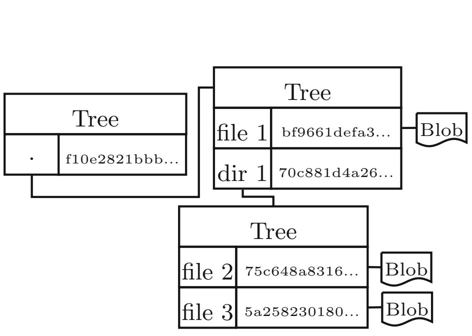
The rest of the paper is organized as follows: Sect. 2 introduces the problem; Sect. 3 discusses the threat model and assumptions that we use; Sect. 4 describes the design goals and the architecture of VMIGuard; Sect. 5 describes the implementation of VMIGuard; Sect. 6 evaluates the performance of VMIGuard; Sect. 7 discusses related work; and Sect. 8 concludes the paper.

2 Background

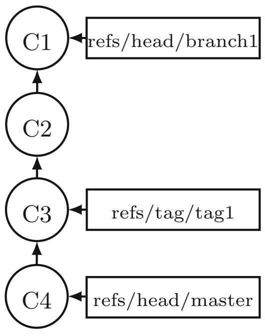
In this section, we first describe the integrity mechanisms that are an integral part of git. Next, we outline several possible integrity attacks that can be performed by a malicious insider. The problem that this paper aims to solve is finding a solution that helps detect and prevent these attacks.

2.1 Git and Its Security

Git is a distributed VCS where every user's working copy of the code is a repository and it holds the whole history of changes [1]. During the initialization,



(a) Tree



(b) Tag and reference

Fig. 1. Git object

the git command creates a folder called *.git* where git stores and manipulates the metadata. Git has four types of objects: blob, tree, commit, and tag. The commit and the tag are the most relevant objects for this paper.

Git stores content in the manner of a simplified UNIX filesystem. All contents are stored in a tree with blobs. A tree corresponds to a directory and a blob corresponds to the file contents (Fig. 1-A).

A commit contains the information of the author, time of the commit, a reference to the previous commit and a reference to the files that are being committed. When users *commit* their changes, they can propagate the changes to other users via a server by invoking two commands: *push* to send the local changes to the remote repository and *pull* or *fetch* to retrieve the changes from the remote repository to the local repository.

A tag is a reference to a commit object. Git stores references in the *.git/refs* folder. References are also used by *branches* where it stores the hash value that points to the particular *commit* (Fig. 1-B). Git stores the reference as plain-text that can be modified directly, which can cause problems that are explained in the next section.

To detect manipulations in a software repository, it is necessary that a client can validate the integrity of the source code and detect unauthorized manipulations. To achieve that, git implements these security mechanisms that make sure that all changes in a repository are logged in the git history [18]:

– Each commit object contains the hash of the commit metadata (commit message, committer, commit date, author and date), tree, and parent. Where the tree is the hash of the working directory and the parent is the hash of the commit before that being modified. Those hashes ensure the integrity of commits.
– Users have the option to sign a commit using GNU Privacy Guard (GPG). It ensures non-repudiation of a commit.
– Users can use a certificate to sign the references. These certificates address man-in-the-middle attacks.

These mechanisms enable a client to trace down all manipulations for each commit back to the first commit of a repository.

2.2 Attack Types

Nevertheless, there are still approaches that allow attackers to manipulate software repositories without violating integrity mechanisms. However, this requires access to the file system that stores the repository. In particular, Torres-Arias et al. [18] describe the following three attack scenarios on git:

- **Teleport Attacks:** modification of a git reference to point to an arbitrary object;
- **Rollback Attacks:** modification of a git branch reference to point to an older commit in that branch;
- **Deletion Attacks:** removal of a branch or a tag references.

We introduce one more type of low-level attacker, **powerful malicious administrator**, who can access the git server with sufficient privilege to add some files to the repository and uses the teleport attack to include the new files into a reference as shown in Fig. 2. First, a benign user sets a tag (V1.0) on the last commit (C3) as shown on Fig. 2-A. Second, the benign user pushes a new commit (C4) ahead of C3 as shown on Fig. 2-B. Lastly, the malicious administrator injects a new commit (E1) from inside the server and modifies the reference of V1.0 tag to point to the malicious commit as shown on Fig. 2-C.

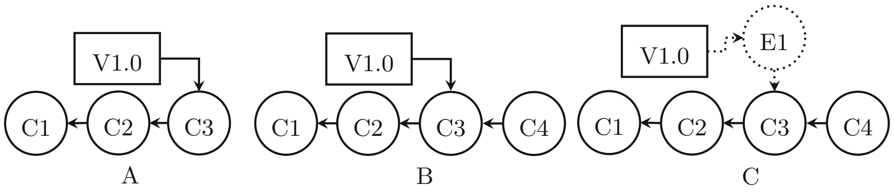


Fig. 2. Powerful malicious administrator attack scenario

3 Threat Model and Assumptions

For this paper, we assume that an attacker has access to the virtual machine that runs the git server and can manipulate the files of the software repositories. This applies, for example, to a malicious insider as well as to an external attacker who exploits software bugs to gain access to the system. We assume that the operating system is trusted and that an attacker does not manipulate the kernel and system call functions.

For our research prototype, we assume that all communication between the server and a client is established via SSH and that the SSH communication is trustworthy. We assume that an attacker can replace the SSH binary used by the git server to bypass the monitoring. A different SSH binary as the original one might contain different data structure and symbols layout that may make the monitoring impossible. We assume that the monitoring and database virtual machines are well isolated where the attacker is unable to intercept and temper the log files.

4 Design Goals and Architecture of VMIGuard

We approach the threats described in the previous sections with our VMIGuard approach that leverages virtual machine introspection for detecting and preventing VCS service integrity violations. This section discusses the design goals and presents the architecture of VMIGuard.

4.1 Goals

From the high-level perspective, VMIGuard aims at achieving the following design goals:

(G1) **Detection:** VMIGuard shall detect all attacks on the git services that have been described in Sect. 2.2.
(G2) **Isolation:** VMIGuard shall not require any in-guest agents (i.e., monitoring software installed on the VCS server itself) so that attackers that have access to that server are not able to modify the extracted data nor manipulate or disable VMIGuard.
(G3) **Persistent logs:** VMIGuard shall store the logs of VCS operations that are necessary for integrity violation checks inside persistent storage using a database solution.
(G4) **Notification:** VMIGuard shall have a mechanism to notify the user that the git repository has been modified or is in a faulty state. VMIGuard shall prevent the propagation of the inconsistent repository.
(G5) **Performance:** VMIGuard shall add only a small overhead to the response time of the VCS.

With those goals in mind, our implementation covers these requirements:

(R1) Extract transferred data (SSH traffic) without any modification on the client-side and the presence of Man-in-the-Middle (MiTM).

(R2) Process the extracted data which is related to git activity over SSH.

(R3) Store specific information of the filtered data based on the specific git activity such as: *commit*, *pull*, or *push*.

(R4) Check the integrity of the git repository that is being accessed by the user.

(R5) Direct notification to the user if a violation happens.

(R6) Produces small overhead on the running system.

(R7) Detect any modification on the SSH process.

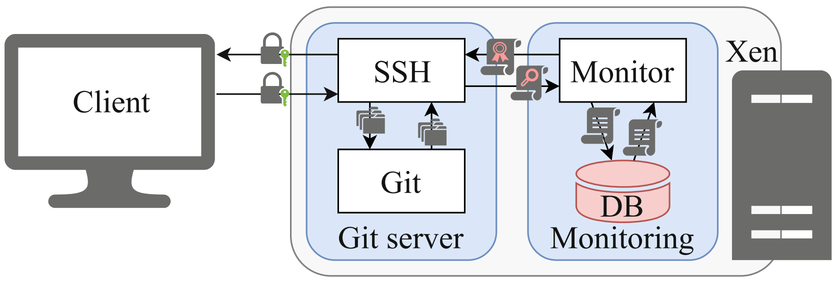


Fig. 3. VMIGuard's architecture

4.2 Components

Figure 3 shows the architecture of VMIGuard. On the left-hand side, there is a standard git client interacting with a git server using an SSH channel (encrypted). Git supports both SSH and HTTPS-based communication. Our approach could similarly be applied to both variants, but in the following, we limit the discussion to the case of using SSH channels. The git server is a standard off-the-shelf installation of git in a virtual machine.

On the right-hand side, there is the monitoring component of VMIGuard. The monitoring component needs permissions to perform virtual machine introspection (VMI) on the git server, and the VMI mechanisms must support active VMI, i.e., enable tracing operations in the monitored system by intercepting the control flow. Our prototype implementation uses the Xen hypervisor and the CloudPhylactor architecture [17], which leverages Xen security modules (XSM) to enable running the VMIGuard monitor in a dedicated virtual machine, with permission to introspect the git server. The git server has no access to the monitoring virtual machine. This allows us to isolate the monitoring VM that holds the database and protect it against malicious insiders.

The connection between the client and the server uses an encrypted and authenticated SSH channel. Due to the encryption, simple network-based monitoring of the interaction between git client and server is not feasible. However, the communication endpoint in the server (the SSH daemon) will decrypt client requests and pass them to the git service, and receive plain-text responses before encrypting and sending them to the client. Our **monitor module** extracts the SSH data from the SSH daemon's memory by leveraging VMI (R1) (R2) and stores the information into a database and for our implementation, we use ElasticSearch [4] (R3). The monitor module reads previous logs (if exist) that related to the repository and produces a decision whether there is an integrity violation or not (R4). If there is a violation, the monitor module changes the payload of the SSH connection in the git virtual machine so that the manipulated git content is not passed to the client. Also, a message is added inside the transmitted SSH packet to notify the user directly about the integrity violation (R5).

VMIGuard uses function tracing of the SSH daemon to intercept the control flow and extract function call input parameters. We use software breakpoints to intercept function calls and implement Xen Altp2m [10] to increase the stealthiness and minimize the overhead of the monitoring $\textcircled{R6}$. We download the debugging symbols of the OpenSSH that is installed on the git server. We use the debugging symbol to extract the symbols' (functions') offsets to calculate the function's memory address and the data structure layout to extract the SSH network packet payload.

VMIGuard checks the integrity of the SSH daemon to ensure that the tracing mechanism is working as intended $\textcircled{R7}$. We periodically read the executable section of the SSH daemon on the memory. If the results differ, we assume that the process has been changed and no longer can be trusted. We also block the network traffic to prevent un-monitored client's activities.

To detect manipulations of an attacker to a git repository, we monitor and log changes of clients via the SSH service to rebuild the state of the repository. When a user requests information of the repository, we crosscheck if the content sent (that is read from the filesystem) matches the state in our database.

5 Implementation

We have implemented a prototype version of VMIGuard. For the git server, we install Ubuntu 16.04 and gitea as a lightweight git frontend. On the monitoring virtual machine, we install LibVMI [11] as a library for basic VMI function. In this section, we provide additional implementation details of our prototype.

5.1 Extracted Information

VMIGuard extracts some information from git client–server interaction. We provide more details on how this extraction process works in Sect. 5.2 below. VMIGuard stores the extracted information into a database (we use ElasticSearch (ES) [4] for the current implementation). The relevant elements that are contained in client–server messages and recorded to the database are the following:

- **repository:** The git repository's name that is being accessed.
- **timestamp:** The time when a particular repository is being accessed.
- **ip:** The IP address of the client who accesses the repository.
- **refs:** The git reference that is being accessed.
- **hash:** The hash value of the latest commit that is uploaded to the server.
- **old hash:** The old hash value of the older commit that is being replaced by the new one.
- **new hash:** The new hash value that replaces the older commit.

5.2 Data Extraction

Git uses either HTTPS or SSH protocol to ensure the security of the transmitted data. Conceptually, the method of this paper works for both protocols but, we use SSH for our implementation.

VMIGuard extracts the data of an SSH connection by tracing two OpenSSH functions, they are: *sshbuf_get_u8* and *ssh_packet_send2_wrapped*. Those functions are responsible for doing the encryption and decryption of a packet. We extract the packet's payload to analyze it before it is encrypted (send) and after it is decrypted (receive).

VMIGuard filters the SSH packets by matching the executed command. If it executes *git-receive-pack*, it means that the server is receiving content that should be pushed into the repository.

```
exec....$git-receive-pack  'stewart/test1.git'
557bd81abf33322115324a379d225d86fcdbd618
    ↪ ad47e374c23280a3024420cfed423dc1e8ff117a  refs/heads/
    ↪ master.
```

Where, *'stewart/test1.git'* is the repository name, *557bd8...* is the old hash, *ad47e3...* is the new hash , and *ref/heads/master* is the refs that being updated. If, it executes *git-upload-pack*, it means that the server is pushing the repository to the user.

```
exec....# git-upload-pack  'stewart/test1.git'
ad47e374c23280a3024420cfed423dc1e8ff117a  refs/heads/master.
```

Where the repository name is the same and *ad47e3...* is the hash commit of ref *refs/heads/master* that is being accessed.

VMIGuard extracts the IP address of the user by accessing the SSH session data structure on the memory. It holds the private session state metadata such as origin IP, origin port, incoming packet, and outgoing packet.

If VMIGuard detects a violation, it performs two actions:

1. Adds a warning message on the SSH's network packet to notify the user.
2. Replaces the transferred files with the empty string which produce an error on the client-side that prevents the user to receive the files.

Those actions will be shown on the git client of the user:

```
Cloning into 'test1'...
remote: WARNING!! inconsistent
fatal: pack signature mismatch
fatal: index-pack failed
```

5.3 Cross Checking – Integrity Violation Detection

When users push changes to their repositories, VMIGuard records all commit hash values and their references accordingly. When a user clones or pulls a repository, VMIGuard checks the last commit hash and compares it against the most recent *commit* hash (R4).

Algorithm 1. Push

```
1: procedure STORE_LOG($repo$,$ref$,$hash$)
2:     if $repo$ exist then
3:         create_new_repo_record($repo$)
4:     end if
5:     store_new_hash_value($repo$,$ref$,$hash$)
6: end procedure
```

Algorithm 2. Pull

```
    procedure IS_VALID($repo$,$ref$,$hash$)
2:      $a \leftarrow false$
        if $repo$ and $ref$ exist then
4:          $temp \leftarrow$ latest stored hash
            if $temp == hash$ then
6:              $a \leftarrow true$
            end if
8:      end if
        return $a$
10: end procedure
```

5.4 SSH Integrity Check

To ensure the integrity of the SSH daemon, VMIGuard analyze the executable section of the process as shown in Fig. 4. First, in an initial training phase, VMIGuard reads the (read-only) address range of the executable section that are mapped in memory. Second, it reads 4 KB each time from the beginning until the end of the section and computes a hash over it. Third, it appends all the hashes and computes the final hash over them. Lastly, it stores the final hash value inside the database.

At normal run-time, the integrity of the SSH daemon is ensured repeatedly by computing the hash of the program and comparing it with the hash stored in the database.

6 Evaluation

We measure the performance impact of VMIGuard. We use three scenarios that simulate normal usage of a git server, they are:

Ⓢ1 Generate one file with x MB in size, push it to the server, and pull the change onto a different folder; where $x = [1, 10]$.

Ⓢ2 Generate two files with x MB in size (for each file), push it to the server, and pull the change onto a different folder; where $x = [1, 10]$.

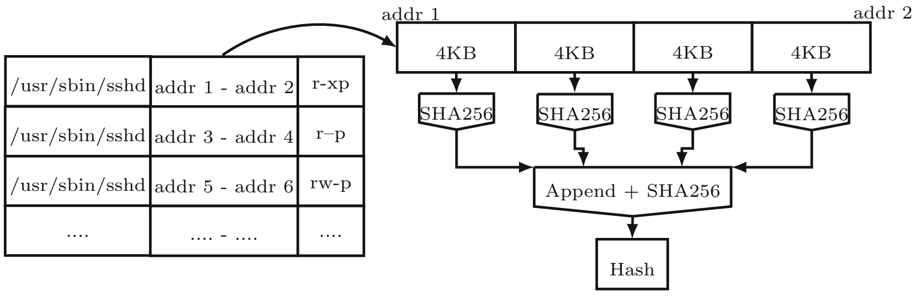


Fig. 4. Executable memory region hashing

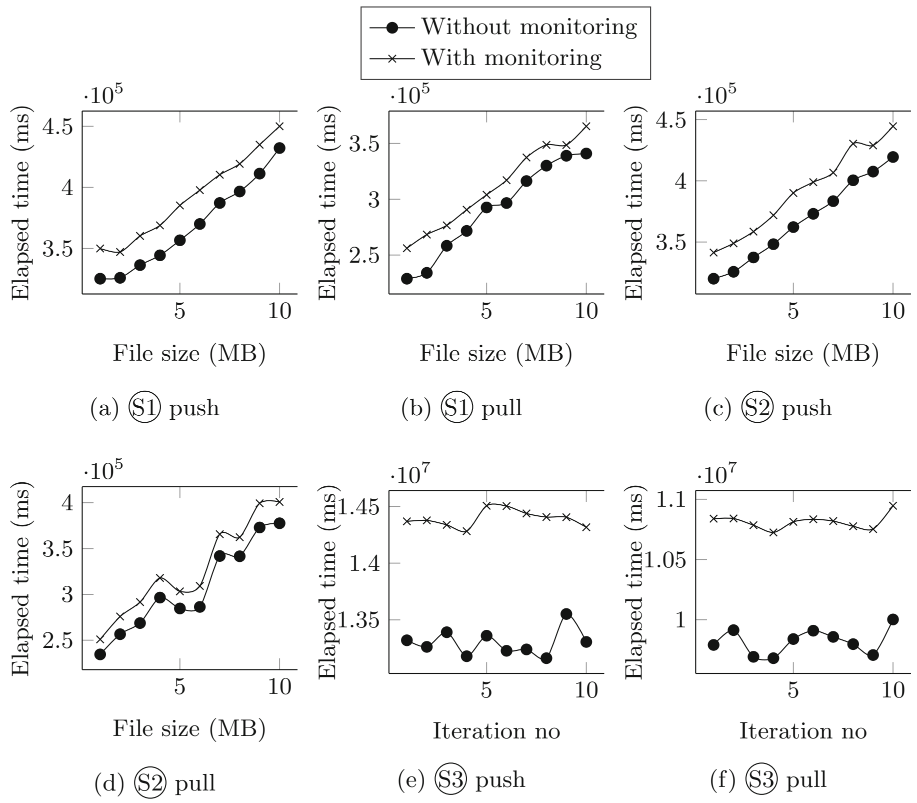


Fig. 5. VMIGuard's performance evaluation

S3 Generate a 50 KB file, push to the server, and pull the change onto a different folder. This is repeated 200 times so that the repository will end up with 200 files.

We execute scenario S1 and S2 100 times, and scenario S3 for ten times then, we calculate the average of elapsed time on the client-side. We run the client and the server on the same host machine to minimize the networking delay. Figure 5 a-d shows the result of scenario S1 and S2 where the overhead in average is 6.5% and 7.1% for push and pull respectively. Figure 5e and f shows the result of scenario S3 where the overhead is 8.2% and 10% for push and pull respectively.

6.1 Limitations

There are several limitations that we do not address in the current version of VMIGuard:

- An adversary that has access to the *dom0* can shut down the monitoring VM.

– The client is not able to detect whether the monitoring is running. Hence, the current version is usable for providers or companies that want to increase their security level against adversaries.
– The current version does not apply encryption for the log, but we assume that the isolation is secure enough that the adversary is not able to eavesdrop or tamper the log.

7 Related Work

Aublin et al. introduce *LibSEAL* [2]. LibSEAL detects integrity violation of a public file services by modifying the TLS library implementation to extract the data, stores the extracted data inside a relational database, with the system protected by a Trusted Execution Environment (TEE) such as Intel SGX. In contrast, VMIGuard does not require any modification of the monitored system and produces lower overhead.

Logging-as-a-service [12,19] leverages a trusted third party (TTP) to maintain the logs. The TTP helps to detect the integrity violation. The user can compare their locally stored logs against the TTP. On the other hand, VMIGuard does not require the presence of a trusted third party. The system owner can decide what to do with the stored data that is generated by VMIGuard.

TLSKex [16] uses VMI to extract TLS sessions keys from the memory with a brute-force (trial and error) approach to determine the key's location in the memory in order to decrypt the communication. Similarly, DroidKex [15] also use brute-force approach but, it learns the data structure layout to reduce the overhead of finding the key's location. Unlike those approaches, VMIGuard does not require the extraction of the key to decrypt the encrypted connection since it directly extracts the plain data in memory.

We introduce *Sarracenia* [13], a VMI based SSH honeypot. *Sarracenia* dumps the SSH data into log files for later analysis. VMIGuard uses the same data extraction mechanism as *Sarracenia*, but VMIGuard analyzes the SSH data on-the-fly and stores only the data that is considered necessary.

8 Conclusions

To detect integrity violation of a git repository, we introduce VMIGuard, a VMI based VCS integrity violation detection. The main strength of this study is the exclusion of needs to do modification of the server. Our implementation shows that VMIGuard produces a small increase in the system's response time. Although this study focuses on git via SSH, the findings may well have a bearing on other protocols. This would be a fruitful area for further work. To support other work, we published the source code on GitHub https://github.com/libvmtrace/libvmtrace.

Acknowledgment. This work has been supported by the German Federal Ministry of Education and Research (BMBF) in the project DINGFEST-EFoVirt and German Research Foundation (DFG) in the project ARADIA.

References

1. Atlassian Bitbucket: What is Git (2018). https://www.atlassian.com/git/tutorials/what-is-git. Accessed 19 July 2019
2. Aublin, P.L., et al.: LibSEAL: revealing service integrity violations using trusted execution. In: Proceedings of the Thirteenth EuroSys Conference. ACM (2018)
3. Bitbucket: Bitbucket (2018). https://bitbucket.org/. Accessed 19 July 2019
4. Elasticsearch B.V.: Open Source Search & Analytics - Elasticsearch — Elastic (2010). https://www.elastic.co/. Accessed 22 July 2019
5. Gitea: Git with a cup of tea, painless self-hosted git service (2018). https://gitea.io/. Accessed 19 July 2019
6. GitHub: GitHub (2018). https://github.com/. Accessed 19 July 2019
7. GitLab: GitLab (2018). https://gitlab.com/. Accessed 19 July 2019
8. Gogs: Gogs is a painless self-hosted Git service (2018). https://gogs.io/. Accessed 19 July 2019
9. Jain, B., Baig, M.B., Zhang, D., Porter, D.E., Sion, R.: SoK: introspections on trust and the semantic gap. In: Proceedings of the 2014 IEEE Symposium on Security and Privacy, SP 2014, pp. 605–620. IEEE Computer Society, Washington, DC (2014). http://dx.doi.org/10.1109/SP.2014.45
10. Lengyel, T.K.: Stealthy monitoring with Xen altp2m. https://blog.xenproject.org/2016/04/13/stealthy-monitoring-with-xen-altp2m/. Accessed 13 July 2019
11. Payne, B.D.: Simplifying virtual machine introspection using LibVMI. Technical report SAND2012-7818, Sandia National Laboratories (2012)
12. Ray, I., Belyaev, K., Strizhov, M., Mulamba, D., Rajaram, M.: Secure logging as-a-service-delegating log management to the cloud. IEEE Syst. J. **7**(2), 323–334 (2013)
13. Sentanoe, S., Taubmann, B., Reiser, H.P.: *Sarracenia*: enhancing the performance and stealthiness of SSH honeypots using virtual machine introspection. In: Gruschka, N. (ed.) NordSec 2018. LNCS, vol. 11252, pp. 255–271. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03638-6_16
14. Software Freedom Conservancy: Git -distributed-is-the-new-centralized (2005). https://git-scm.com/. Accessed 19 July 2019
15. Taubmann, B., Alabduljaleel, O., Reiser, H.P.: DroidKex: fast extraction of ephemeral TLS keys from the memory of Android apps. Digit. Investig. **26**, S67–S76 (2018)
16. Taubmann, B., Frädrich, C., Dusold, D., Reiser, H.P.: TLSkex: harnessing virtual machine introspection for decrypting TLS communication. Digit. Investig. **16**, S114–S123 (2016)
17. Taubmann, B., Rakotondravony, N., Reiser, H.P.: CloudPhylactor: harnessing mandatory access control for virtual machine introspection in cloud data centers. In: Trustcom/BigDataSE/ISPA, 2016, pp. 957–964. IEEE (2016)
18. Torres-Arias, S., Ammula, A.K., Curtmola, R., Cappos, J.: On omitting commits and committing omissions: Preventing git metadata tampering that (Re)introduces software vulnerabilities. In: USENIX Security Symposium, pp. 379–395 (2016)
19. Zawoad, S., Dutta, A., Hasan, R.: Towards building forensics enabled cloud through secure logging-as-a-service. IEEE Trans. Dependable Secur. Comput. **1**, 1 (2016)

Correction to: MicroSCOPE: Enabling Access Control in Searchable Encryption with the Use of Attribute-Based Encryption and SGX

Antonis Michalas, Alexandros Bakas, Hai-Van Dang, and Alexandr Zalitko

Correction to:
Chapter "MicroSCOPE: Enabling Access Control in Searchable Encryption with the Use of Attribute-Based Encryption and SGX" in: A. Askarov et al. (Eds.): ***Secure IT Systems*****, LNCS 11875,**
https://doi.org/10.1007/978-3-030-35055-0_16

The original version of this chapter contained an error in the fourth author's name. The spelling of Alexandr Zalitko's name was incorrect in the header of the paper. The author name was corrected.

The updated version of this chapter can be found at
https://doi.org/10.1007/978-3-030-35055-0_16

Author Index