

Sokratis K. Katsikas · Frédéric Cuppens
Nora Cuppens · Costas Lambrinoudakis
Christos Kalloniatis · John Mylopoulos
Annie Antón · Stefanos Gritzalis (Eds.)

LNCS 10683

Computer Security

ESORICS 2017 International Workshops
CyberICPS 2017 and SECPRE 2017
Oslo, Norway, September 14–15, 2017
Revised Selected Papers

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7410>

Sokratis K. Katsikas · Frédéric Cuppens
Nora Cuppens · Costas Lambrinoudakis
Christos Kalloniatis · John Mylopoulos
Annie Antón · Stefanos Gritzalis (Eds.)

Computer Security

ESORICS 2017 International Workshops
CyberICPS 2017 and SECPRE 2017
Oslo, Norway, September 14–15, 2017
Revised Selected Papers


Editors

Sokratis K. Katsikas 
Norwegian University of Science
and Technology
Gjøvik
Norway

Frédéric Cuppens
IMT Atlantique
Brest
France

Nora Cuppens
IMT Atlantique
Brest
France

Costas Lambrinouidakis
University of Piraeus
Piraeus
Greece

Christos Kalloniatis 
University of the Aegean
Mytilene
Greece

John Mylopoulos
University of Toronto
Toronto, ON
Canada

Annie Antón
Georgia Institute of Technology
Atlanta, GA
USA

Stefanos Gritzalis
University of the Aegean
Karlovassi
Greece

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-72816-2 ISBN 978-3-319-72817-9 (eBook)
<https://doi.org/10.1007/978-3-319-72817-9>

Library of Congress Control Number: 2017962874

LNCS Sublibrary: SL4 – Security and Cryptology

© Springer International Publishing AG 2018, corrected publication 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

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



Preface

This book contains revised versions of the papers presented at the Third Workshop on Security of Industrial Control Systems and Cyber-Physical Systems (CyberICPS 2017) and the First International Workshop on Security and Privacy Requirements Engineering (SECPRE 2017). Both workshops were co-located with the 22nd European Symposium on Research in Computer Security (ESORICS 2017) and were held in Oslo, Norway, on September 15, 2017.

CyberICPS aims to bring together researchers, engineers, and governmental actors with an interest in the security of industrial control systems and cyber-physical systems in the context of their increasing exposure to cyber-space, by offering a forum for discussion on all issues related to cyber-security. Cyber-physical systems range in size, complexity, and criticality, from embedded systems used in smart vehicles, to SCADA and industrial control systems like energy and water distribution systems, smart transportation systems etc.

CyberICPS 2017 attracted 32 high-quality submissions, each of which was assigned to three referees for review; the review process resulted in ten full and two short papers being accepted to be presented and included in the proceedings. These cover topics related to threats, vulnerabilities, and risks that cyber-physical systems and industrial control systems face; cyber attacks that may be launched against such systems; and ways of detecting and responding to such attacks.

For many years, software engineers have focused on the development of new software thus considering security and privacy mainly during the development stage as an ad hoc process rather than an integrated one initiated during the system design stage. However, the data protection regulations, the complexity of modern environments such as IoT, IoE, cloud computing, big data, cyber-physical systems etc. and the increased level of users' awareness in IT have forced software engineers to identify security and privacy as fundamental design aspects leading to the implementation of more trusted software systems and services. Researchers have addressed the necessity and importance of implementing design methods for security and privacy requirements elicitation, modeling, and implementation in the past few decades. Today security by design (SbD) and privacy by design (PbD) are established research areas that focus on these directions. SECPRE aimed to provide researchers and professionals with the opportunity to present novel and cutting-edge research on these topics.

SECPRE 2017 attracted 14 high-quality submissions, each of which was assigned to three referees for review; the review process resulted in accepting five papers to be presented and included in the proceedings. These cover topics related to security and privacy requirements assurance and evaluation, and to security requirements elicitation and modeling.

We would like to express our thanks to all those who assisted us in organizing the events and putting together the programs. We are very grateful to the members of the Program Committees for their timely and rigorous reviews. Thanks are also due to the Organizing Committees for the events. Last, but by no means least, we would like to thank all the authors who submitted their work to the workshops and contributed to an interesting set of proceedings.

November 2017

Sokratis K. Katsikas
Frédéric Cuppens
Nora Cuppens
Costas Lambrinoudakis
Christos Kalloniatis
John Mylopoulos
Annie Antón
Stefanos Gritzalis

Organization

CyberICPS 2017 General Chairs

Nora Cuppens	IMT Atlantique, France
Costas Lambrinouidakis	University of Piraeus, Greece

CyberICPS 2017 Program Committee Chairs

Sokratis K. Katsikas	Norwegian University of Science and Technology - NTNU, Norway; and University of Piraeus, Greece
Frédéric Cuppens	IMT Atlantique, France

CyberICPS 2017 Technical Program Committee

Cristina Alcaraz	University of Malaga, Spain
Samiha Ayed	Devoteam, France
Mauro Conti	University of Padua, Italy
Mourad Debbabi	Concordia University, Canada
Hervé Debar	Telecom SudParis, France
David Espes	University of Brest, France
Joaquin Garcia-Alfaro	Telecom SudParis, France
Dieter Gollmann	Technical University of Hamburg, Germany
Wael Kanoun	Nokia, France
Jean Leneutre	Telecom ParisTech, France
Javier Lopez	University of Malaga, Spain
Masahiro Mambo	Kanazawa University, Japan
Sjouke Mauw	University of Luxembourg, Luxembourg
Weizhi Meng	Institute for Infocomm Research, Singapore
Chris Mitchell	Royal Holloway, UK
Jonathan Petit	Security Innovation, USA
Juha Röning	University of Oulu, Finland
Yves Roudier	EURECOM, France
Pierangela Samarati	Università degli Studi di Milano, Italy
Radu State	University of Luxembourg, Luxembourg
Houbling Song	West Virginia University, USA
Craig Valli	Edith Cowan University, Australia
Jozef Vyskoc	VAF, Slovakia
Khan Ferdous Wahid	Airbus Defence and Space GmbH, Germany
Stephen Wolthusen	Royal Holloway, UK
Stefano Zanero	Politecnico di Milano, Italy

CyberICPS 2017 Additional Reviewers

Abbas Acar
Saed Alrabae
Nathan Clarke
Pallavi Kaliyar
Marcello Pogliani
Mario Polino
Sarada Prasad
Davide Quarta
Paria Shirani

SECPRE 2017 General Chairs

Annie Antón	Georgia Institute of Technology, USA
Stefanos Gritzalis	University of the Aegean, Greece

SECPRE 2017 Program Committee Chairs

John Mylopoulos	University of Toronto, Canada
Christos Kalloniatis	University of the Aegean, Greece

SECPRE 2017 Technical Program Committee

Travis Breaux	Carnegie Mellon University, USA
Frédéric Cuppens	Telecom Bretagne, France
Sabrina De Capitani di Vimercati	Università degli Studi di Milano, Italy
Theo Dimitrakos	University of Kent, UK
Mohamad Gharib	University of Florence, Italy
Paolo Giorgini	University of Trento, Italy
Maritta Heisel	University of Duisburg-Essen, Germany
Jan Juerjens	University of Koblenz-Landau, Germany
Costas Lambrinouidakis	University of Piraeus, Greece
Tong Li	Beijing University of Technology, China
Javier Lopez	University of Malaga, Spain
Fabio Martinelli	National Research Council, C.N.R., Italy
Aaron Massey	University of Maryland, USA
Haralambos Mouratidis	University of Brighton, UK
Liliana Pasquale	University College Dublin, Ireland
Michalis Pavlidis	University of Brighton, UK
David Garcia Rosado	University of Castilla-La Mancha, Spain
Mattia Salnitri	University of Trento, Italy
Pierangela Samarati	Università degli Studi di Milano, Italy
Nicola Zannone	Eindhoven University of Technology, The Netherlands

SECPRE 2017 Additional Reviewers

Francesco Mercaldo
Mina Sheikhalishahi

Contents

Protecting Industrial Control and Cyber-Physical Systems

Towards End-to-End Data Protection in Low-Power Networks	3
<i>Vasily Mikhalev, Laurent Gomez, Frederik Armknecht, and José Márquez</i>	
Development of an Embedded Platform for Secure CPS Services	19
<i>Vincent Raes, Jan Vossaert, and Vincent Naessens</i>	
Introducing Usage Control in MQTT.	35
<i>Antonio La Marra, Fabio Martinelli, Paolo Mori, Athanasios Rizos, and Andrea Saracino</i>	

Threats, Vulnerabilities and Risks

Towards Security Threats that Matter	47
<i>Katja Tuma, Riccardo Scandariato, Mathias Widman, and Christian Sandberg</i>	
A Methodology to Assess Vulnerabilities and Countermeasures Impact on the Missions of a Naval System.	63
<i>Bastien Sultan, Fabien Dagnat, and Caroline Fontaine</i>	
STRIDE to a Secure Smart Grid in a Hybrid Cloud	77
<i>Bojan Jelacic, Daniela Rosic, Imre Lendak, Marina Stanojevic, and Sebastijan Stoja</i>	

Cyber Attacks in Industrial Control and Cyber-Physical Systems

Stealthy Deception Attacks Against SCADA Systems	93
<i>Amit Kleinmann, Ori Amichay, Avishai Wool, David Tenenbaum, Ofer Bar, and Leonid Lev</i>	
On Ladder Logic Bombs in Industrial Control Systems	110
<i>Naman Govil, Anand Agrawal, and Nils Ole Tippenhauer</i>	
Enforcing Memory Safety in Cyber-Physical Systems	127
<i>Eyasu Getahun Chekole, John Henry Castellanos, Martín Ochoa, and David K. Y. Yau</i>	

Detecting Attacks in Industrial Control and Cyber-Physical Systems

Supporting the Human in Cyber Defence 147
Kirsi Helkala, Benjamin J. Knox, Øyvind Jøsok, Ricardo G. Lugo, Stefan Sütterlin, Geir Olav Dyrkolbotn, and Nils Kalstad Svendsen

CRBP-OpType: A Constrained Approximate Search Algorithm for Detecting Similar Attack Patterns 163
Ambika Shrestha Chitrakar and Slobodan Petrović

Multistage Downstream Attack Detection in a Cyber Physical System 177
Rizwan Qadeer, Carlos Murguia, Chuadhry Mujeeb Ahmed, and Justin Ruths

Security and Privacy Requirements Assurance and Evaluation

A UML Profile for Privacy-Aware Data Lifecycle Models 189
Majed Alshammari and Andrew Simpson

Evaluation of a Security and Privacy Requirements Methodology Using the Physics of Notation 210
Vasiliki Diamantopoulou, Michalis Pavlidis, and Haralambos Mouratidis

Security Requirements Elicitation and Modelling

What Users Want: Adapting Qualitative Research Methods to Security Policy Elicitation. 229
Vivien M. Rooney and Simon N. Foley

An Anti-pattern for Misuse Cases 250
Mohammad Torabi Dashti and Saša Radomirović

Decision-Making in Security Requirements Engineering with Constrained Goal Models. 262
Nikolaos Argyropoulos, Konstantinos Angelopoulos, Haralambos Mouratidis, and Andrew Fish

Erratum to: Enforcing Memory Safety in Cyber-Physical Systems. E1
Eyasu Getahun Chekole, John Henry Castellanos, Martín Ochoa, and David K. Y. Yau

Author Index 281

Protecting Industrial Control and Cyber-Physical Systems

Towards End-to-End Data Protection in Low-Power Networks

Vasily Mikhalev^{1(✉)}, Laurent Gomez², Frederik Armknecht¹,
and José Márquez³

¹ University of Mannheim, Mannheim, Germany
{mikhalev,armknecht}@uni-mannheim.de

² SAP Product Security Research, Mougins, France
laurent.gomez@sap.com

³ SAP IoT and Industrie 4.0, Walldorf, Germany
jose.marquez@sap.com

Abstract. An important, emerging trend in the context of the Internet of Things (IoT) are low-power networks (LPNs), referring to networks that target devices with very limited access to energy sources. While there are several approaches that allow to comply to these novel power restrictions, none of them provide a sufficient level of security, in particular with respect to data protection.

In this paper, we propose a data protection scheme that ensures end-to-end security from low-power devices to backend applications. It meets the technical constraints imposed by LPNs, while preserving data confidentiality and integrity. Our solution has been deployed on the water distribution network of the City of Antibes in France. The evaluation of the overhead introduced by the proposed data protection scheme shows promising results with respect to power (battery) consumption.

Keywords: Internet of Things · Low-power networks
End-to-end security · Data protection · Key management
Constrained devices · Edge computing

1 Introduction

1.1 Motivation

The Internet of Things (IoT) is seen as one of the most ground breaking and game-changing evolutions of operational and information technology in modern times. More and more enterprises are nowadays aiming to connect both new and legacy physical assets to their system landscapes in order to capture the data from these assets, generate insights and derive value out of the latter. This requires to retrofit existing physical assets in order to leverage them as part of the (connected) physical (IoT) infrastructure.

A major part of this growing amount of “things” (devices) is expected to be low-powered, i.e., devices which are restricted to consume only very little energy

to operate (and therefore to communicate). This has numerous consequences in practice; for instance, one cannot expect these devices to hold an active link but rather to communicate on-demand only. To this effect, these devices will have to communicate not only with a reduced packet size, but to embrace both a higher latency and a lower throughput at run-time. This takes us to the concept of Low-Power Connectivity, materialized as Low-Powered Wide-Area Networks (LPWANs) or Low-Power Networks (LPNs). LPNs offer an economically viable option to physically deploy new sensors along with the necessary communications infrastructure in order to generate, transport and ingest data coming from any type of asset. This means that part of the great potential of the LPNs rely on the cost effectiveness of retrofitting “old” assets with new (low-power) sensors and (low-power) connectivity, making this type of approach the first choice when targeting legacy assets and landscapes.

However, even if the connectivity is achieved, security is often an equally important requirement - in particular end-to-end data protection from the devices (i.e. the first end) all the way to the backend applications (i.e. the second end). The involvement of multiple actors in an IoT scenario (e.g. device, network, platform, application, professional services providers, etc.) together with LPN constraints makes the fulfillment of an end-to-end data protection (i.e. confidentiality and integrity) a challenging endeavor.

In fact, to the best of our knowledge, none of the existing LPN technologies provide real end-to-end confidentiality and integrity of the data, neither in the sense of complying to the NIST recommendations for the proper use of cryptography primitives. For example the SIGFOX technology provides no encryption at all per default [18]. In LoRaWAN AES encryption is deployed but it uses a single key [19]. That is, once the key is set, it is never updated, which may reduce the security level due to the fact that certain attacks become easier when a larger amount of data is collected applying the same key. The same issue also persists in the ZigBee networks [11].

In order to increase security in IoT environments over the last few years several key-establishment protocols have been proposed [1, 2, 12–17]. All these protocols can be divided into two main categories: two-party communication key management protocols and group communication key management protocols.

The philosophy in the first category is to create a common session key between two parties. Usually these type of protocols require to use asymmetric cryptographic schemes which is not feasible in the context of LPNs. There exist several approaches [1, 13, 14, 16] which allow to use the 3rd party (i.e. edge) compute power to alleviate the load on the constrained devices. All these protocols have rather high communication complexity which is undesirable in the low-power wide-area networks, since it considerably increases the energy consumption, and also some of them rely on the security of a 3rd party, implying by consequence that those are not providing end-to-end security.

The overall idea of the protocols of the second category, e.g. see [2, 12, 15, 17], is that the key establishment is done among the group of three or more entities. Therefore, these protocols do not really serve for providing end-to-end

security between the device nodes and the central application node. If one node is compromised, the entire group is compromised. Moreover, this type of schemes have a rather high communication complexity.

In addition all protocols mentioned above require that the devices natively support bi-directional communication, which is not always possible in existing LPN technologies.

1.2 Contribution

In this paper we describe a solution which allows us to solve both problems at the same time: achieving connectivity between the legacy assets and the enterprise systems, while guaranteeing end-to-end data protection. We propose a data protection scheme to be embedded into the application level in order to guarantee data confidentiality and authenticity, following the NIST recommendations [3, 7, 9, 10]

For this purpose we consider two different approaches for encryption: a more “classical” approach by using AES in counter mode and a more recent approach using format preserving encryption [10]. The latter is useful in the cases when it is desirable that plaintext and ciphertext possess the same structure to avoid modifications of the existing databases when inserting encrypted data in the same tables where previously unencrypted data was persisted.

We demonstrate the suitability of our approach by implementing it to secure the sensor data of the water distribution network of the City of Antibes.

1.3 Structure

The paper is structured as follows. We start by describing the considered use case in Sect. 2. In Sect. 3 we provide the full specifications of the proposed data protection scheme. Section 4 describes the reasons behind the different choices made during the design phase. In Sect. 5 we discuss the security of the scheme and in Sect. 6 we assess its concrete implementation and performance. Section 7 concludes the paper.

2 Use Case

2.1 Description

The City of Antibes, France, instruments 300 kms of water pipelines with 2000+ sensors that capture a variety of data, e.g. debit, temperature, pressure, water storage levels. The collected information is sent over an ultra-narrow band network, and pushed to a central application (a predictive maintenance dashboard), depicted in Fig. 1. The latter implements a hydraulic model of the network with the purpose of predicting disruptions in the water supply of the city. This application enables the City of Antibes to optimize their operational budget and to better allocate its work force (as well as external contractors) on the field.

Given that the sensor data carries critical information for a number of operational levels of the city, it is of the utmost importance to meet specific security requirements, being confidentiality and integrity of the sensor data the top priority.

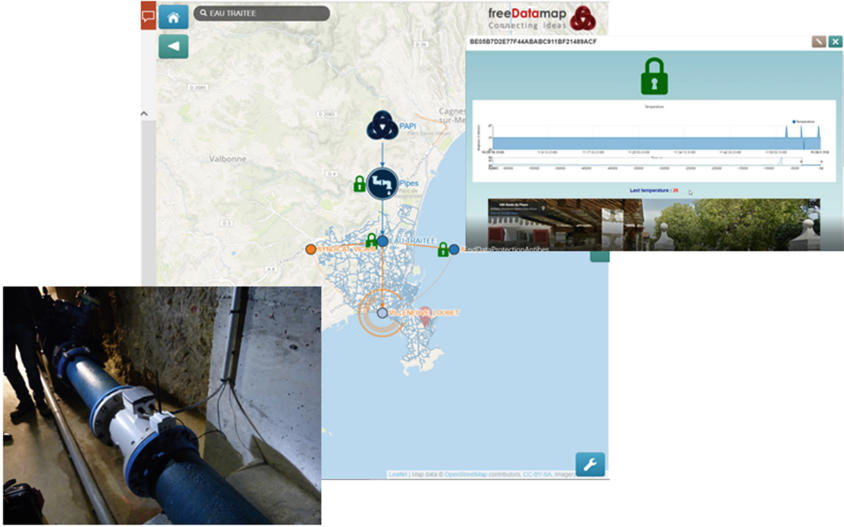


Fig. 1. Predictive maintenance dashboard, city of Antibes

2.2 Security Requirements

The city of Antibes expressed explicitly several security requirements such as end-to-end (e2e) confidentiality of data, i.e., the content of the data needs to be concealed all the way from the time the data is generated (device) to the time the data consumed (application/dashboard). This requirement is driven by the need to be compliant to the EU General Data Protection Regulation [5]. A further requirement stated by the city of Antibes is that the integrity of the data has to be ensured as it strongly determines or predicts the state of the water distribution network. For practical reasons, it is also required that part of the encrypted data is of the same format as the plaintext data.

Based on these discussions and own experience, we came up with the following list of security requirements for our solution:

- It has to guarantee end-to-end confidentiality, authenticity, and integrity of the data.
- All components have to follow NIST recommendations i.e. [3, 7, 9, 10].
- It has to be applicable to different existing low-power networks, even if the maximal payload size is as small as 12 bytes (e.g. SIGFOX).
- It needs to be deployable on the low-power devices.

- It must be applicable in scenarios which are not (yet) supporting bidirectional communication.
- It must be compliant with different encryption algorithms while meeting the compute power restrictions of constrained devices.

Note that while these requirements cover the specific needs of the city of Antibes, they actually apply to a much broader class of scenarios and use cases.

3 The Proposed Solution

Within this section, we propose a concrete solution that fulfills the security requirements formulated in Sect. 2.2. We first explain the underlying reference architecture and describe afterwards the data protection scheme.

3.1 Reference Architecture

As depicted in Fig. 2, our reference architecture is organized end-to-end i.e., from the edge to the backend. Edge refers to those devices that are not under direct control of the backend system. Within the edge, a device connected to the actual sensors provides an entry point of data into the landscape. At the edge, devices have three basic data services: acquisition, protection, and communication of the data toward either the edge gateway or directly to the backend system. Gateways are in this context the bridges between rather constrained devices and the backend systems, enabling the required connectivity. The protection of sensor data is realized by a *Data Protection Service*. The data is sent from sensors over a LPN to a low-power gateway which then forwards it to the backend.

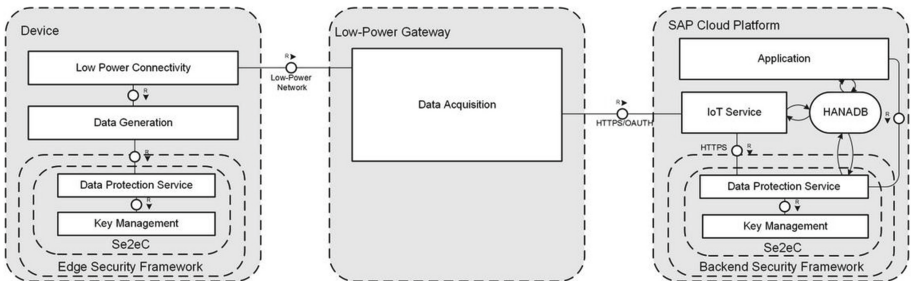


Fig. 2. Reference architecture, secure end-to-end communications

On the backend, the central application uses this *Data Protection Service* to check the validity of the protected data. That is, it detects and reports any attempts for data injection or replay attacks. Once data validity is confirmed, the data is decrypted.

On both, the device and backend, the *Data Protection Service* delegates the task of key management to the *Key Management* component. In our architecture, the device stores only one master key that is used for enabling both encryption and authentication. On the backend, the *Key Management* maintains a list of the master keys of all the devices involved.

3.2 Data Protection Scheme

Next we provide the full specification of the proposed data protection scheme. The used notation is summarized in Table 1. In a nutshell, the scheme can be divided into two parts: the key management part and the data protection part. As explained above, the scheme builds on master keys that are pre-shared between each of the devices and the backend application. With respect to key management, the scheme generates single-use keys which are applied to encrypt and authenticate every new packet to be sent. To achieve synchronization, it implements intermediate keys and sequence numbers.

The encryption and authentication processes are independent from each other. In order to keep the communication complexity low, we are using format preserving encryption algorithms where the ciphertext size equals the plaintext size, e.g. AES in counter mode [7] and FF1 [10]. Authentication is realized by computing a message authentication code (MAC) of the cipher text.

Data Protection Scheme on the Device. Here we describe the operations performed on each of the devices. The pseudo code is given in Algorithm 1.

Each device stores its device ID ID and a master key K_{master} in protected (access controlled) memory. These values are set during the deployment of the devices into a landscape and are known to the central application. The master key is used implicitly for both, encryption and authentication, by deriving appropriate intermediate keys from the master key; more precisely, the data sent to the backend is divided into packets and those are protected by applying different keys. Packets are grouped in *sequences* of length SN_{max} each. Our scheme uses a different intermediate key K_i for each sequence. When ever the sequence number reaches SN_{max} , a new sequence is started using a new intermediate key. This is handled at steps 1–7.

Moreover, the device keeps track of the sequence number, referring to the number of the packet in the current sequence. For each packet within a given sequence, an encryption key K_{Enc} and an authentication key K_{MAC} are derived from the current intermediate key K_i . The required keys are computed at steps 8–9.

Then the message is encrypted and the ciphertext is authenticated. The payload to be sent is composed of the ciphertext C , authentication tag T , sequence number SN , and device ID ID .

Data Protection Scheme on the Backend. Next we explain the operations which take place on the backend. To distinguish between the values received from

Table 1. Notation

Key management	
K_{master}	Master key. A 128-bit key preshared between the user application and the device This key is never changed and is used to generate intermediate keys
i	Intermediate key index. An integer value used for computing the next intermediate key This value is updated by being incremented by 1 every SN_{max} times
K_i	Intermediate key. A 128-bit key generated from the current intermediate key index and the master key, used for generating encryption and authentication keys
K_{Enc}	Encryption key. A 128-bit key used for encrypting the data
K_{MAC}	Message authentication code key. A 128-bit key used for computing message authentication code tags
Data	
M	Message. The sensor value that has to be transmitted from the node to the backend application
C	Ciphertext
T	Authentication tag
Meta data	
ID	Device ID. The unique identifier of a device
SN	The sequence number of the packet generated using the current intermediate key
Algorithms	
$Enc(K_{Enc}, M)$	An algorithm that encrypts a message M using a secret key K_{Enc} and produces a ciphertext C
$Dec(K_{Enc}, C)$	An algorithm that decrypts a ciphertext C using the secret key K_{Enc} and outputs a plaintext message M
$MAC()$	An algorithm that computes a message authentication code for some data using authentication key K_{MAC}
$CMAC()$	Cipher-based message authentication code [9]
Parameters	
SN_{max}	The maximum value for the sequence number. Represents the maximum number of packets to be processed using the same intermediate key K_i
L	Length of the authentication tag T
n^{max}	The maximal number of invalid packets with the same meta-data (ID, SN) for current intermediate key index i
Backend stored values	
\mathcal{S}^i	The list of sequence numbers already used with the same key index i for a given device
n_{SN}^i	The number of packets with the same meta-data (ID, SN) for the intermediate key index i currently received by the backend side
$Decrypted[ID, SN, i]$	Boolean value which is true if a packet with the given meta-data (ID, SN) for intermediate key index i was already decrypted

Prerequisites:

- encryption scheme $\text{Enc}()$;
- master key K_{master} ;
- intermediate key index i ;
- sequence number SN ;
- Device Id ID .

Input : Plaintext message M

Output : Payload PL

```

1 if ( $SN \neq SN_{max}$ ) then
2 |  $SN \leftarrow SN + 1$ ;
3 else
4 |  $SN \leftarrow 0$ ;
5 |  $i \leftarrow i + 1$  ;
6 |  $K_i \leftarrow \text{CMAC}(K_{master}, i)$ 
7 end
8  $K_{Enc} \leftarrow \text{CMAC}(K_i, SN \parallel ID \parallel 0)$  ;
9  $K_{MAC} \leftarrow \text{CMAC}(K_i, SN \parallel ID \parallel 1)$  ;
10  $C \leftarrow \text{Enc}(K_{Enc}, M)$ ;
11  $T \leftarrow \text{CMAC}(K_{MAC}, C)$ ;
12  $PL \leftarrow (C \parallel T \parallel SN \parallel ID)$ ;
13 Return( $PL$ );

```

Algorithm 1: Data protection scheme: encryption and authentication on the constrained device side

the nodes and those ones which are computed, we use the upper indexes rec and com respectively. For example, T^{rec} denotes the value of the authentication token in a received packet, while T^{com} denotes the token computed at the backend side.

Let $K_{master}, i, S^i, n_{SN}^i$ be the values used by the device with identifier ID . The scheme is shown in Algorithm 2. At steps 3–10, it is checked if packets with the same meta data were already processed by the algorithm, if one of them was already verified and decrypted $\text{Decrypted}[ID, SN, i]$, or if the number of attempts exceeded the maximum n^{max} . In all these cases the algorithm returns corresponding errors. Otherwise, at steps 14–17 the authentication tag is verified. In case that the tag is valid the decryption process begins.

Encryption Algorithms. We consider two different variants of encryption algorithms, which allow to preserve the size¹ of the plaintext while computing the ciphertext. These are:

- AES encryption in counter mode [7]:

$$C = \text{Enc}(K_{Enc}, M) \quad (1)$$

$$= M \oplus \text{AES}_{128}(K_{Enc}, IV^0) \quad (2)$$

where IV^0 is a 16-byte zero vector:

¹ This is required to adapt the packet size to possible length restrictions.

Prerequisites: For each of the devices which communicate with the application:

- encryption scheme $\text{Enc}()$;
- master key K_{master} ;
- current intermediate key index, i ;
- all sequence numbers \mathcal{S}^i already used for current intermediate key index i
- information if corresponding packets were already verified and decrypted $\text{Decrypted}[ID, SN, i]$;
- the number of messages received with the same sequence number n_{SN}^i

Input : Payload PL ; length of MAC L

Output : Device ID ID , Sequence number SN , Message M , or error

```

1  ( $C^{rec}, TAG^{rec}, SN, ID$ )  $\leftarrow PL$  // splitting payload
2  Use  $K_{master}, i, \mathcal{S}^i, n_{SN}^i$  for the device ID =  $ID$ ;
3  if ( $SN \in \mathcal{S}^i$ ) then
4  |   if ( $\text{Decrypted}[ID, SN, i]$ ) then
5  |   |   Return(Error: replay attack detected);
6  |   end
7  |    $n_{SN}^i \leftarrow n_{SN}^i + 1$ ;
8  |   if ( $n_{SN}^i > n^{max}$ ) then
9  |   |   Return(Error: too many attempts);
10 |   end
11 else
12 |    $\mathcal{S}^i \leftarrow \mathcal{S}^i \cup \{SN\}$ .
13 end
14  $K_i \leftarrow \text{CMAC}(K_{master}, i)$ ;
15  $K_{MAC} \leftarrow \text{CMAC}(K_i, SN \parallel ID \parallel 1)$ ;
16  $T^{com} \leftarrow \text{CMAC}(K_{MAC}, C^r)$ ;
17 if ( $T^{rec} \neq T^{com}$ ) then
18 |   Return(Error: Injection detected);
19 else
20 |    $K_{Enc} \leftarrow \text{CMAC}(K_i, SN \parallel ID \parallel 0)$ ;
21 |    $M^{com} \leftarrow \text{Dec}(K_{Enc}, C^{rec})$ 
22 end
23  $\text{Decrypted}[ID, SN, i] \leftarrow true$ ;
24 if  $SN = SN_{max}$  then
25 |    $i \leftarrow i + 1$ 
26 end
27 Return( $M$ );

```

Algorithm 2: Data protection scheme: authentication check and decryption at the backend side

- the format preserving encryption scheme $FF1_{enc}$ [10].

$$C = \text{Enc}(K_{Enc}, M) \quad (3)$$

$$= FF1_{enc}(K_{Enc}, TW, M) \quad (4)$$

The tweak TW is computed as the concatenation of the device ID and sequence number: $TW = (ID \parallel SN)$.

4 Design Criteria

The goal was to design a scheme that meets the requirements in Sect. 2.2.

4.1 Key Management

Master Key. The master key is a preshared 128-bit secret value which has to be stored in the secure memory of the device. It has to be properly generated for each of the unique device identifiers. From the master key all the other keys are derived. We note that most of the existing LPN technologies/providers offer a possibility to equip the devices with master keys.

Key Generation Approach. For the generation we are using the cipher-based message authentication code CMAC [9], as it is recommended by the NIST for secure key generation from preshared secrets [3].

Intermediate Keys. Intermediate keys are used together with sequence numbers to make sure that no two messages are encrypted and/or authenticated using the same secret keys K_{Enc} , K_{MAC} .

Sequence Number. The reasons to include a sequence number are the following. First of all, it allows each time to generate different authentication and encryption keys, even under the same intermediate key. Due to this property, we do not need to update the intermediate key index and to write to the non-volatile memory every time, which is a rather energy consuming operation. Moreover, the sequence number is included into the payload to increase the reliability of the communication. For example, if a packet is lost or delayed, the backend will immediately get this information from the value of the sequence number of the next packet. Since the sequence number has to be included into the payload, we want to keep its size as small as possible. For example, in our current setting we use 16-bit long sequence numbers. This allows us to achieve both: update the intermediate key only rarely, while reducing to the outmost the increase of the payload.

Frequency of Intermediate Keys Update. Every SN_{max} times of being used, the intermediate key is updated. The value SN_{max} depends on the length of the sequence number. For example, if the sequence number is 16 bit long, we set SN_{max} to be equal to 65535. This is done in order to avoid the situation when the same (i, SN) pair are used twice on the same device.

In addition, when the device is reset², the intermediate key index is incremented and the sequence number is zeroed before any other computations begin. This allows that pairs (intermediate key/sequence number) are only used once.

Frequency of Encryption and Authentication Keys Update. We use new encryption and authentication keys for every packet to be sent. This ensures that an attacker cannot collect multiple data connected to the same keys. Moreover, even if one of the packets and its corresponding keys are compromised, this does not hint to get information about the other packets. Although, it may seem that such approach would require a lot of computational effort, our experiments show that the scheme is efficient as demonstrated in Sect. 6.

4.2 Authentication and Encryption

Different Keys for Authentication and Encryption. A standard solution for achieving authentication and encryption with symmetric cryptography is to use CCM mode as recommended by the NIST [8]. It would probably be more efficient compared to the solution presented in this paper because it allows to achieve authentication and encryption using the same key for both. However, this mode is not flexible and does not support different encryption algorithms, contradicting our agnostic principles. As we need that the data protection scheme is compatible with other encryption schemes such as FF1, we realize authentication and encryption separately based on different single-time keys.

Encrypt Then Authenticate. We first encrypt and then apply a chosen MAC on the encrypted data rather than computing MAC of the plaintext. This option provides the integrity of both the plaintext and the ciphertext [4]. If the ciphertext is wrong, it is filtered out immediately and there is no need for decryption.

AES in Counter Mode. In most of the real-life cases we use AES in CTR mode to have short payload and low computational complexity (one call of AES per packet for sensor data size below 128-bit). We note that this mode is among the ones recommended by the NIST [7]. We keep the initialization vector constant IV^0 since we change the encryption key K_{Enc} with every new packet.

Format Preserving Encryption Scheme $FF1_{enc}$. For the cases when it is required that the ciphertext not only has the same size as the plaintext but

² Resetting the device may be easily done by an attacker, e.g., by interrupting the power supply or even by using a hard-reset button, which is available/accessible on most devices.

also has to be stored in the same format on the backend side, we are using the FF1 algorithm recommended by the NIST [10]. In comparison with the second recommended alternative called FF3, FF1 is more flexible as it accepts more formats of the input data and uses size tweaks. Moreover, there are recent indications of weaknesses in the FF3 algorithm [6].

Tweak Generation for $FF1_{enc}$. The main recommendation for the FF1 tweak generation [10] is that it should vary with each instance of the encryption as much as possible. Note that, the tweak has to be associated with the given plaintext and doesn't necessarily need to be secret. To fulfill these requirements, we use the meta-data of a given packet, which is associated with each plaintext per se, and generate the tweak by concatenating the sequence number and the device ID.

Choice of Message Authentication Codes. We follow the NIST recommendations [9] and use CMAC for the authentication.

5 Security

5.1 Attacker Model

In our security model we assume that an attacker has full access to the communication channel between the device and the backend. That is, an attacker can eavesdrop all the exchanged messages and modify them freely. We also give an attacker the possibility to reset a device as many times as needed. However, we do not consider side-channel attacks, as these depend on concrete devices and implementations.

5.2 Authentication of the Sender

Each message to be sent by the device contains a cryptographic token (authentication tag) that is computed based on the single-time used authentication key. Relying on the NIST [9] we assume that without knowing the authentication key the chances of an attacker to apply the forgery attack against a given packet with the same meta-data (ID, SN for current i) is not higher than $2^{-L} \cdot n^{max}$.

To keep the payload size low, in our most lightweight implementation we consider L to be as low as 32 bits and fix n^{max} to 1, meaning that if a packet with the given meta-data is rejected, all the other similar packets are rejected without further verification. This leads to a risk of 2^{-32} which is sufficient for our desirable security level. However, we note that increasing the length of the tag would allow to achieve higher security levels for the cost of higher communication complexity.

5.3 Data Integrity

Similar discussions apply to the integrity of the message. If it is corrupted or changed it will be accepted as a valid one only with probability 2^{-L} .

5.4 Data Confidentiality

Both encryption schemes that we discuss in the paper, namely AES 128 in counter mode and the format preserving encryption FF1 were selected strictly following the recommendations by the NIST [7, 10]. Relying on these, we assume that there are no attacks with a complexity lower than 2^{128} which would break any of these encryption schemes.

5.5 Replay Attacks

Each message contains a sequence number which is verified by the backend in order to discard replay attempts. If the message contains a sequence number which is repeated for the same intermediate key index and the same device, the replay attack is detected. We recall that the described data protection scheme excludes the possibility of using the same pair (key index, sequence number) twice.

5.6 Side-Channel Attacks

The security of the scheme with respect to side-channel attacks depends on the concrete implementation. Therefore, no general arguments can be given about vulnerability. We use primitives from the TinyCrypt library where certain generic side-channel attack countermeasures are implemented³.

6 Evaluation

6.1 Data Protection Service Implementation

For reading the data from the sensors and applying protection mechanisms, our nodes are equipped with Intel Quark C1000 Microcontroller Units (MCUs) connected through LoRaWan modules. At the backend side we are using a decryption service implemented in Java and deployed in the SAP Cloud Platform.

We evaluate the performance of the proposed data protection scheme on Intel Quark C1000⁴, an Intel Microcontroller Unit (MCU). This MCU is equipped with 8 KB of cache, 32 MHz clock speed, 80 KB SRAM, and 384 KB integrated Flash. In our experiments the MCU was powered with 5.07 V.

As depicted in Fig. 2, data is collected by the C1000, protected using the *Data Security Service*, and are sent over LoRa to a gateway. On C1000, the *Data Security Service* is implemented in C, hosted on ZephyrOS⁵. We use cryptographic primitives from its TinyCrypt library.

³ See <http://zephyr-docs.s3-website-us-east-1.amazonaws.com/online/dev/crypto/tinycrypt.html>.

⁴ See <http://www.intel.com/content/www/us/en/embedded/products/quark/mcu/se-soc/overview.html>.

⁵ See <https://www.zephyrproject.org/>.

Protected data is then forwarded to the SAP Cloud Platform⁶. On the backend, the *Data Security Service*, implemented in Java, checks the validity of the protected data against replay attacks and injections. Once the validity of the protected data is confirmed, data is decrypted by the *Data Security Service*.

6.2 Evaluation

In this section, we discuss the overhead on battery (power consumption), memory, and time introduced by the data protection scheme. We consider the overhead on a cloud backend as negligible, as long as the resources are theoretically unlimited there. Our evaluations have been conducted for the two mentioned encryption algorithms: AES in counter mode, and FF1.

Battery and Time Consumption. In Table 2, we summarize the results on the battery and time consumption over 10k temperature data. We distinguish between three steps in this evaluation: (i) data acquisition, (ii) data protection, and (iii) data transmission over LoRaWAN. At data acquisition, we read temperature data from the sensor attached to the C1000. At data protection, we protect the data using the proposed scheme. We have here two implementations of encryption: AES in counter mode and FF1. At data transmission over LoRaWAN, we send data through the C1000 LoRa built-in module to an Intel LoRa gateway.

Table 2. Battery and CPU performance on 10 K data

	Data acquisition	Data protection		LoRa communication
		AES-CTR	FFX	
Battery	<1 mAh	3 mAh	4 mAh	16 mAh
CPU	13.44 s	46.16 s	69.44 s	68 s

On 10k data the overall process takes 150.88s, and consumes 21 mAh with FF1, and 127.6s and 20mAh with AES-CTR. The impact of data protection scheme with AES in counter mode is overall 36.17% on time and 15% on battery consumption. In case of FF1, the overall impact is 45.06% on time and 19.04% on battery consumption.

Memory Consumption. Regarding the total flash memory consumption, the data protection scheme occupies 0.56% while 15.49% is reserved for the cryptographic libraries, TinyCrypt, and 1.56% for LoRaWAN communication.

⁶ See <https://cloudplatform.sap.com/>.

6.3 Discussion

Overall, the estimated overhead incurred on time and battery is at most 33%. The flash memory footprint overhead is negligible.

For a regular 16750 mAh battery, 6.700k sensor data can be acquired and then encrypted with AES in counter mode and sent over LoRaWAN. It allows to process 12 years of data when data is sent every minute, or 190 years of data if sent every 15 min.

7 Conclusion

In this paper, we proposed a Secure End-to-End Communications data protection scheme for low-power devices. Our solution provides data confidentiality and integrity from (IoT) devices to central backend applications by relying on established cryptographic schemes and frequent key updates. Moreover, it respects the technical constraints imposed by low-power devices (e.g. CPU, memory) and low-power connectivity (e.g. high latency, low throughput).

Our solution has been implemented on industrial IoT devices and deployed on the water distribution network of the City of Antibes. The results show an overhead of the data protection scheme on the overall battery consumption of four times less than required for the transmitting the data over LoRaWAN.

As future work, we aim at processing on encrypted data on the edge (recall the reference architecture explained in Sect. 3.1). As of today, the backend is considered as the single source of truth and unique access point. However, we see a paradigm shift towards a distribution of the data and decentralization of the processes into the edge. In that context, there is a clear need for processing data while preserving its confidentiality and integrity.

Acknowledgement. Mr. Patrick Duverger, CIO for the French Government in the City of Antibes, as well as Dr. Steffen Schulz, Intel Collaborative Research Institute for Secure Computing; whose insights and expertise greatly enriched this scientific work.

References

1. Abdmeziem, M.R., Tandjaoui, D.: A cooperative end to end key management scheme for E-health applications in the context of internet of things. In: Garcia Pineda, M., Lloret, J., Papavassiliou, S., Ruehrup, S., Westphal, C. (eds.) ADHOC-NOW 2014. LNCS, pp. 35–46. Springer, Heidelberg (2014)
2. Abdmeziem, M.R., Tandjaoui, D., Romdhani, I.: A decentralized batch-based group key management protocol for mobile internet of things (DBGK). In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), pp. 1109–1117. IEEE (2015)
3. Barker, E., Roginsky, A.: SP 800–133. Recommendation for cryptographic key generation. NIST Special Publication 800:133 (2012)

4. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_41
5. European Commission. EU General Data Protection Regulation (2017)
6. Betül Durak, F., Vaudenay, S.: Breaking the ff3 format-preserving encryption standard over small domains. Cryptology ePrint Archive, Report 2017/521 (2017). <http://eprint.iacr.org/2017/521>
7. Dworkin, M.J.: SP 800–38A. Recommendation for Block Cipher Modes of Operation: Methods and Techniques. National Institute of Standards & Technology (2001)
8. Dworkin, M.J.: SP 800–38C. Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality. National Institute of Standards & Technology (2004)
9. Dworkin, M.J.: SP 800–38B. Recommendation for block cipher modes of operation: The CMAC mode for authentication. National Institute of Standards & Technology (2005)
10. Dworkin, M.J.: SP 800–38G. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. National Institute of Standards & Technology (2016)
11. Lennvall, T., Svensson, S., Hekland, F.: A comparison of WirelessHART and ZigBee for industrial applications. In: IEEE International Workshop on Factory Communication Systems, WFCS 2008, pp. 85–88. IEEE (2008)
12. Li, Y.: Design of a key establishment protocol for smart home energy management system. In: 2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), pp. 88–93. IEEE (2013)
13. Naoui, S., Elhdhili, M.E., Saidane, L.A.: Enhancing the security of the IoT LoRaWAN architecture. In: 2016 International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN), pp. 1–7, November 2016
14. Porambage, P., Braeken, A., Kumar, P., Gurtov, A., Ylianttila, M.: Proxy-based end-to-end key establishment protocol for the Internet of Things. In: 2015 IEEE International Conference on Communication Workshop (ICCW), pp. 2677–2682. IEEE (2015)
15. Renugadevi, N., Swaminathan, G., Kumar, A.S.: Key management schemes for secure group communication in wireless networks—a survey. In: 2014 International Conference on Contemporary Computing and Informatics (IC3I), pp. 446–450. IEEE (2014)
16. Saied, Y.B., Olivereau, A.: D-HIP: a distributed key exchange scheme for HIP-based Internet of Things. In: 2012 IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–7. IEEE (2012)
17. Shen, J., Moh, S., Chung, I.: A novel key management protocol in body area networks. In: Proceedings of the Seventh International Conference on Networking and Services (ICNS 2011), pp. 246–251 (2011)
18. SigFox: Make things come alive in a secure way (2017). https://www.sigfox.com/sites/default/files/1701-SIGFOX-White_Paper_Security.pdf
19. LoRa Alliance Technical Marketing Workgroup. Technical Overview of LoRa and LoRaWAN (2015)

Development of an Embedded Platform for Secure CPS Services

Vincent Raes^(✉), Jan Vossaert, and Vincent Naessens

MSEC, imec-DistriNet, KU Leuven, Technology Campus Ghent,
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
{vincent.raes,jan.vossaert,vincent.naessens}@kuleuven.be

Abstract. Cyber-Physical Systems are growing more complex and the evolution of the Internet of Things is causing them to be more connected to other networks. This trend, combined with the fact that increasingly powerful embedded devices are added to these systems opens up many new opportunities for the development of richer and more complex CPS services. This, however, introduces several new challenges with respect to the data and software managed on these CPS devices and gateways. This paper proposes a platform for the development of secure cyber-physical devices and gateways. The platform provides a secure environment in which critical CPS services can be running. The secure environment relies on the ARM TrustZone security extensions. A commodity Android environment is provided in which the user can install additional software components to extend the functionality of the devices. A prototype of the platform is developed and this prototype is evaluated.

Keywords: TrustZone · Genode · Android · CPS · Security
Embedded system

1 Introduction

The Internet of Things and its integration in Cyber-Physical Systems (CPS) is causing a major evolution in our lives. Previously simple isolated systems are becoming more connected and are equipped with more powerful processors to run more complex services. These new levels of connectivity and processing power provide opportunities for the development of new products and services and, potentially also new business models. Currently IoT/CPS devices are typically integrated in a closed system and serve a specific use case. However, third-party developers could use the hardware capabilities (e.g. sensors, actuators and networking connectivity) to realize additional services. Hence, a more open software model could greatly increase the value of CPS devices by integrating them into additional ecosystems. This, however, introduces several new challenges [9] with respect to the data and software managed on these CPS devices and gateways. This is typically solved by enforcing access control to the data via a back-end system. However, in many cases it is desirable that the software running on the

device itself or the gateway can be extended and the data can be directly accessed locally. Hence, a set of core security sensitive services should be run in a secure environment, isolated from the software components provided by third-parties. These core services provide the base functionality of the device. Software developers can use these core services of the device to develop their own applications and integrate the IoT device in new ecosystems. Existing embedded platforms do not provide the required support to develop these new services. The operating systems typically running on embedded devices (e.g. Windows ME or a Linux variant) make it difficult to sufficiently ensure the required level of security. The size of these OSs code base offers a large attack surface and makes it hard to ensure that there are no bugs which can be exploited on these systems, especially with a more open software ecosystem.

This paper proposes a platform that facilitates the development of these specific types of IoT devices and gateways. Our contributions in this paper are the following. First we propose a platform to provide secure services in a CPS, based on TrustZone security technology. The platform provides built-in support for the development of secure CPS services. These services run isolated from the commodity OS that allows the installation of additional third-party applications on the platform. This platform offers isolated execution for sensitive services and a transparent execution for application that are not security-sensitive. Then, a prototype of the platform is developed which is evaluated on various points such as performance, security and ease of use.

The remainder of this paper is structured in 6 parts. Section 2 reviews other approaches to provide a Trusted Execution Environment (TEE). In Sect. 3, background regarding the used technologies TrustZone and Genode is provided. The design of the platform is discussed in Sect. 5. Section 6 handles the realization of the prototype. The evaluation can be found in Sect. 7 and our results are concluded in Sect. 8.

2 Related Work

This section provides an overview of other approaches which provide security in an untrusted environment. There have been many research endeavours that are focused on protecting applications from an untrusted environment. In this section, three different categories are distinguished, namely software-based, custom hardware and TrustZone-based solutions.

The first category uses hypervisors and software-based protection to offer an isolated execution environment. AppGuard [17] takes advantage of hardware virtualization support such as Intel VT-x and AMD-V and a Trusted Platform Module (TPM) to secure user applications from the OS. It requires no modifications to be made to the application or the untrusted OS. The dependency on a TPM and a virtualization extension make it impractical to use in embedded devices. Other approaches such as InkTag [5] aim to ensure that the OS is functioning correctly. This is done with a hypervisor and a new technique called *paraverification* in which the OS helps verifying its own behaviour. InkTag incurs

a high performance cost since it encrypts and decrypts user processes based on context switches which is not ideal in an industrial setting.

The second category of security solutions consists of custom hardware solutions. Sancus [11, 16] is a hardware platform that is specifically designed to run software of mutually distrusting parties on a single device, while each party has a strong assurance that its software runs untampered. The security offered is completely hardware-based. This solution is highly geared towards networked microprocessor-based devices. We believe there is a place for more powerful embedded devices with a general purpose OS in a CPS setting.

The final category uses TrustZone as a trusted computing technology. SPROBES [4] is focused on the correct execution of an untrusted OS. It enforces that the kernel can only run code from approved code pages. There are cases that require a more in-depth security for their applications. For example in some cases the code that is executed should be kept secret. Correct functioning of the OS does not guarantee code confidentiality. Another TrustZone based solution is the Trusted Language Runtime (TLR) [12, 13]. TLR protects the security sensitive parts of .NET applications by isolating it from the OS and other applications. It is mainly geared towards mobile applications and especially Windows Phone systems. It also protects a specific part of a single application while we wish to offer services that are available to multiple untrusted applications.

3 Preliminaries

3.1 ARM TrustZone

ARM TrustZone is a set of hardware security extensions implemented on most recent ARM processors. The security extensions allow the device to run in two different processor modes, a Normal and a Secure mode. Each mode represents a virtual processor, providing two isolated execution environments. As the name suggests, the Secure mode is typically used to run security critical software components, isolated from less trusted code running in the Normal mode. The division between Secure and Normal mode is not limited to the CPU but is propagated over the system bus to peripheral devices and memory controllers, providing system-wide security. Access to specific peripherals can be restricted to Secure-mode software. A process running in the Normal mode is unable to access the Secure-mode memory regions or use Secure-mode peripherals. It can also be used to trap specific interrupts to the Secure world. Code running in the Secure mode is capable of reaching Normal-mode resources. The collection of resources accessible to software running in Normal/Secure mode is called the Normal/Secure world. The creation of two virtual worlds on a single core is achieved with a new processor bit, the NS-bit. The world in which the processor is running is determined by this bit. The system bus and memory controllers use the NS-bit to check whether a specific resource request is permitted in the context of the active world.

Since the processor can only execute in one security mode at a time, a way to switch worlds is required to run software in both security modes. A dedicated

processor instruction, the Secure Monitor Call (smc) can be used to activate the Secure world from the Normal world. The instruction changes the processor's operating mode and starts executing the Secure Monitor. The Secure Monitor is part of the Secure world and responsible for controlling communication between both worlds. Based on arguments provided with the Secure Monitor Call, the Monitor can invoke specific software components in the Secure world. The secure world can directly change the operating mode of the processor after it has finished executing the required software components and, subsequently, pass control back to the Normal world.

3.2 Genode

Genode [2] is a framework for the development of highly secure micro Operating Systems. It has been selected as the OS to run in Secure world of the platform presented in this paper. As a microkernel, Genode avoids pitfalls of regular kernels by stepping away from a monolithic kernel structure. The various services that are usually managed by the central kernel, such as device drivers and the file system, run in separate sandboxes. Genode extends the microkernel idea by de-composing the system policy as well as the operating system code. This is achieved by imposing a organizational structure on each part of the system.

A Genode application can be allowed to contact other applications with several Genode interfaces. These are a Remote Procedure Call (RPC) interface and an asynchronous notification interface. These methods are sufficient for calling functions from another service but are not suited for sending large chunks of data. The RPC messages are bound to a small size and the asynchronous notifications have no additional payload. If large amounts of data need to be propagated, Genode allows processes to share some of their assigned memory.

4 Problem Statement

With this paper we wish to propose a solution for the increased security needs in CPS and IoT environments. Recently these systems have become more connected to other networks and are using more powerful embedded devices. Since these systems were typically isolated they are lacking in terms of security. Our proposal consists of a platform that hosts secure services in a secure environment while still allowing untrusted third party applications on the device. The untrusted applications run in a familiar environment that is easily extensible. Developers of secure services get tools to create and add secure services on the platform, and a way to contact their services from the untrusted world is available. The secure world is transparent for the untrusted applications so if an application does not utilize a service, it is unaffected by their existence.

4.1 Requirements

Secure Execution: A trusted execution environment is offered in which services can run isolated from the untrusted rich OS. This environment guarantees the integrity and privacy of the service.

Secure Boot: Secure boot is used to guarantee the authenticity of the secure world and secure services upon startup.

Secure Service Development: The platform provides a build environment for the development and deployment of secure services. Secure services typically use cryptography to support authentication or confidentiality of data. This should, hence, be included in the build environment. Most services will also communicate with some kind of remote endpoint, e.g. a server or a sensor. Thus, the build environment should offer a way to set up a secure tunnel with these remote endpoints.

Rich Normal World: There should be a simple way for new applications to be installed and used in the normal world. The normal world should be able to efficiently run user applications. Application developers should be able to easily develop applications to run in the normal world and use secure world services.

4.2 Attacker Model

The attacker model used for this paper assumes the following capabilities for the attacker. First, the attacker is capable of controlling all the software running in the normal world. This is due to the normal world OSs large code base which makes it prone to exploits. Any applications running in the secure world and the secure world implementation itself are assumed to be trustworthy.

Second, any communication channels to and from the device are assumed to be controlled by the attacker. This allows the attacker to perform Man-in-the-Middle attacks, sniff the network and modify any traffic on the channel.

Further, the attacker is assumed to be unable to break cryptographic primitives, but can perform protocol-level attacks. This follows the Dolev-Yao [1] attacker model.

Finally, TrustZone's attacker model is used for hardware attacks. This means that the attacker is capable of performing low level hardware attacks such as using debugging tools but advanced attacks such as usage of an electron microscope are out of scope.

5 Design

The platform relies on the TrustZone security technology to realise a secure world in which security sensitive services are running, and a flexible normal world system which is managed by the user. This section first presents the architecture of the platform which includes the software stack, the inter-world communication, and the boot process. Subsequently, the software development support provided by the platform is discussed.

5.1 Platform Architecture

Software Stack. The platform has two separate software stacks running on it. The normal world environment runs the Android operating system. Android provides a familiar environment for users in which new applications, developed by third party developers, can be easily installed. It also offers a rich UI which has been geared towards touchscreens. The secure world runs the Genode OS. The secure services running on the platform are separate Genode applications. Each service implements a single Genode RPC function. By calling this function, the service is executed. If large chunks of data need to be passed to a service Genode’s memory sharing technique, described in Sect. 3, is used. Apart from the secure services, the secure monitor is also running as a Genode application in the secure world. The secure monitor manages a list of references to the different services running in the secure environment. Each service in the list is linked to a unique 8-byte identifier. When the monitor is called with one of these identifiers, the corresponding service will be contacted. Figure 1 shows a graphical representation of the software on the platform.

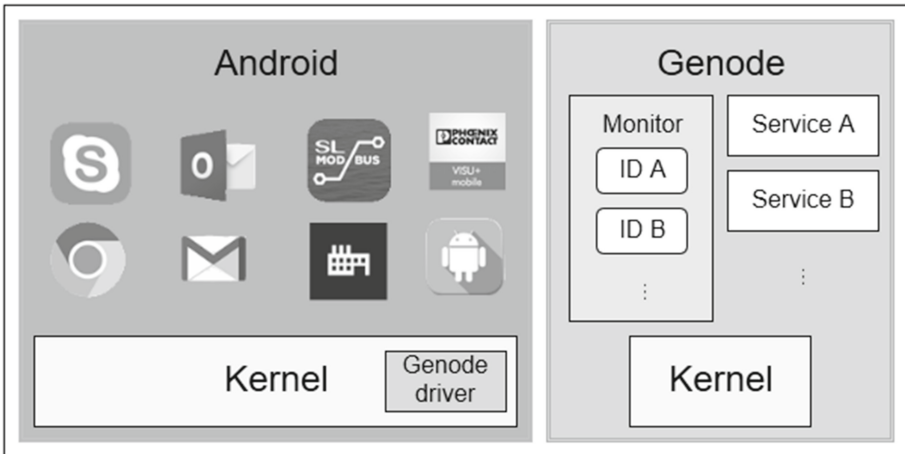


Fig. 1. The software on the platform

World Switch and Inter-world Communication. The secure world behaves as a slave to the normal world. When a normal world application requires data from a secure world service, it issues a request to the secure world. To enable these requests to the secure world, a new driver was added to the Android platform. The driver is responsible for executing the `smc` command which triggers the world switch. Furthermore, the driver manages the data structures that are passed to and from the secure world. A virtual device interface is provided to higher layers of the Android stack. This interface requires the service identifier and one data structure containing all the input for the requested service.

Once the `smc` command is executed, the TrustZone hardware passes control to the secure monitor service. When the secure monitor receives control, it reads the identifier of the requested service and calls the RPC function of that service. The data structure is also passed to the service. Once a service has finished the requested function, execution is returned to the monitor along with the generated result. The monitor then writes the response to the Android driver's memory and switches back to the normal world. Finally the driver responds to the calling function with the data written by the secure world.

Platform Boot. A TrustZone enabled board starts in TrustZone's secure mode. This ensures that the non-secure software is incapable of interfering with the execution of software when the device is booting. The boot process in our platform starts with a secure world bootloader. This bootloader loads the Genode image into memory and checks the integrity of the image using a hash value signed by a key that is saved in the board's hardware. If the authentication is successful, Genode boots as the secure OS. Once the TrustZone configuration is complete and the secure services have been started, Genode operates as bootloader for the normal world. It copies the Android image into memory and prepares the CPU registers accordingly. When the setup has been completed, the first world switch occurs and Android starts operating.

5.2 Software Development Support

This section further discusses the support provided by the platform to develop both applications that use services provided in the secure world and the secure services itself.

Android Application Development. Developers can use the regular development environment provided by Google to develop their Android applications. A Java library is provided to enable communication with secure services. The library offers a generic communication API that allows applications to specify the identifier of the secure service which should be contacted and any additional data for the service. In order to access the previously discussed Android driver, the library uses JNI. Developers of secure services can build on top of this library to provide a service-specific interface on top of the generic communication library. This service-specific library can automatically fill in the service identifier and serialize the different parameters for the service.

Secure Service Development. For the development of secure services, the Genode framework and external C/C++ libraries can be used. The Genode framework itself allows programs to be developed in C and C++. Genode provides a `libc` implementation based on FreeBSD's `libc` which is fairly complete. Genode's `libc` implementation can be extended using plugins to provide additional functionality.

External libraries can be ported to the Genode framework to provide advanced or specialised functions. Porting is a matter of providing the necessary

functionality, if necessary through patching the library, and ensuring libraries the new library depends on, such as `libc`, are available. In our platform a port of the OpenSSL 1.0.1u library is provided since many secure services require cryptographic operations. This OpenSSL library is used as the base for another library which helps setting up secure tunnels to remote endpoints. Secure services can use either of these libraries to use the functions they offer.

6 Prototype

A prototype of the platform has been developed using the i.MX6 SABRE Lite board. This section discusses the hardware-specific TrustZone modifications of the Secure and Normal world software. Subsequently, the use of the secure boot features of the i.MX6 SABRE Lite in the platform is discussed. The code used for this prototype can be found at <https://bitbucket.org/vincent.raes/>.

6.1 Secure World Software

The Genode OS has been modified to make use of TrustZone’s security capabilities. To achieve this, drivers were written for the various TrustZone peripherals. Every driver activates one part of TrustZone’s security measures. The security measures for memory, peripherals and interrupts each have a separate driver. Our platform has the following TrustZone peripherals: the Central Security Unit (CSU), the TrustZone Address Space Controller (TZASC) and the Generic Interrupt Controller (GIC) [3].

The CSU is used to determine what peripherals can be accessed by each world. It assigns various security levels to the board’s peripherals. These security levels vary from allowing secure and normal world processes to reach the peripheral to extremely strict policies in which only privileged processes in the secure world are capable of accessing the device. Since our platform aims to be as transparent towards Android as possible with few modifications to the Android kernel, only the peripherals that manage TrustZone settings are set to be exclusive for the secure world.

The TZASC is the default ARM device to split memory between the secure and normal world. It divides the available memory into various zones which can be assigned different security levels. These security levels are similar to those the CSU offers for peripherals. We require secure memory for our Genode runtime so a section of the available memory is set to secure world only. The size of this section is determined by the build configuration.

Finally the GIC is used to assign interrupts to either world. This is accomplished by using the two separate interrupt types ARM offers: the IRQ and the Fast IRQ (FIQ). It is possible to configure a type of interrupt so it is linked to one world. The GIC assigns an interrupt type to the interrupts available on the platform based on their identifier. Due to our goal of transparency for the normal world, there are no secure world only interrupts.

6.2 Building the Software

The boot image used on the platform is built in two stages. The first stage consists of building the Android system. Android has been patched so it can run side-by-side with the secure world. The patch disables access to any resources that are reserved for the secure world. This is achieved by modifying the build configuration file and the Device Tree Blob (DTB). Additionally, the patch adds our new driver to the Android build. This driver reserves 1024 KB which is used to communicate with the secure world.

The second stage is building the Genode system. Genode has been configured to only occupy limited resources on the hardware. Unlike a commodity OS such as Android, Genode requires knowledge of the system when it is being built. The applications that will run on the system are determined prior to the build process, since new applications cannot be installed or updated when the system is running.

The configuration for both build processes is performed in one global config file. This includes options such as the available memory for each world and what secure services will run on the system. A script is provided which uses the options in this file and builds the system accordingly. The script is used to launch the build scripts for Android and Genode. The Android build process results in the Android kernel and userland. The kernel image is reused during Genode's build process where it is packaged with the Genode kernel. The Genode output consists of a single boot image in which the kernel, the secure services, and the Android kernel are packaged. In order to determine what secure services need to be built and linked to the monitor, the global configuration file is used. Based on the selected services, placeholders found in the Genode code and scripts will be modified to dynamically build and link the required services.

6.3 System Boot

The boot process consists of several steps. The first step uses the boot ROM library and the High Assurance Boot (HAB) library to provide Encrypted boot [14] for the bootloader. The availability of encrypted boot and these software libraries is dependent on the used hardware platform. The software libraries are unchangeable and can be seen as trusted software components. Encrypted boot allows a device to be configured so only authenticated code which has been encrypted can run on the device.

Figure 2 shows the boot process. When Encrypted boot is enabled, the boot ROM loads the bootloader image to memory. This image contains the encrypted bootloader and digital signature data and public key certificate data. The additional data is collectively called the CSF data. The image is generated off-line using a dedicated tool. Encrypted boot also adds the key which has been used to encrypt the code to this image. The stored key has been encrypted with a hardware key unique for every CPU core. This results in an encrypted boot image which can only be executed by one board since the stored key is unique per CPU.

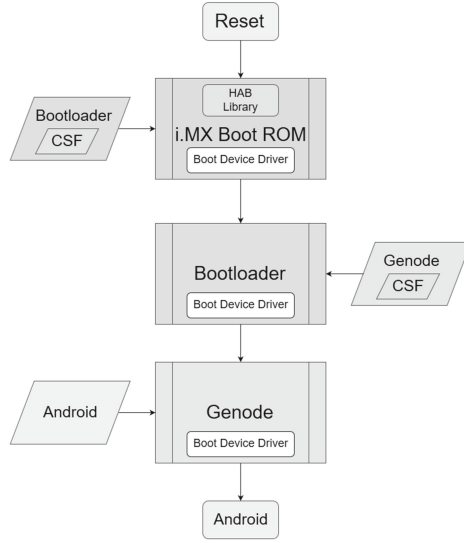


Fig. 2. The secure boot process

With the image completely loaded into memory, the HAB library takes over. This library verifies the signatures and decrypts the image. The loaded code can only be executed if the verification succeeds.

The second step starts once the bootloader has passed verification and has started. The bootloader loads the Genode image to memory. The Genode image contains the encrypted Genode kernel and the additional data used for verification. The bootloader can then issue a command which passes control to the HAB library to verify and decrypt the Genode image.

In the third step Genode acts as the bootloader for Android. Genode loads Android into memory and prepares it for booting by setting the DTB, correcting the CPU registers and setting the Instruction Pointer to the start of the Android kernel. Once this is done, control over the device is passed to the Normal World and Android boots as the final boot step.

7 Evaluation

7.1 Requirements Review

Secure Execution. In order to evaluate the security of the platform we focus on the attack vectors available to adversaries. The biggest attack vector is the Secure Monitor Call. The interface offered by the call is rather narrow. The smc interface is only used to contact secure services. A secure service has limited influence in the secure world due to Genode’s security mechanisms.

Secure Boot. Secure boot in the platform is achieved using both board specific technology and the TrustZone model. The Encrypted boot technology is used

to ensure the authenticity of the secure world bootloader, which in turn verifies the authenticity of the secure world image. This ensures the trustworthiness of the critical steps in boot process.

Secure Service Development. Writing a secure service is a relatively simple matter. Services are programs written in C++ in which at least one RPC call is made available. Based on the input a service receives with this RPC call, various functions can be executed. Genode’s configuration and code needs to be appropriately modified for every service made available but the global build script is capable of automating this process so developers can focus solely on the functionality of their service.

To provide an idea regarding the performance of the cryptographic library, several test have been executed on the prototype. Table 1 show the results of timing measurements of symmetric (i.e. AES) and asymmetric (i.e. RSA) cryptographic operations on both the Android platform and on the secure world. The measurements were made with the Performance Monitor Unit. This is a reliable counter accessibly by both the normal and secure world. The RSA implementation used in Genode is clearly very inefficient compared to the implementation Android uses. Genode is on average 40 times slower in a single operation. The results of the AES implementation on the other hand show a slight delay but the difference isn’t quite as large as with RSA. The fact that Genode is slower has multiple reasons. First, OpenSSL on Genode does not support using assembly optimization for the used platform. Further, Genode does not take advantage of the cryptographic accelerator the hardware offers.

Table 1. Time for cryptographic operations

Type of operation	Genode (ms)	Android (ms)
RSA public	53,6	1,4
RSA private	1774,1	41,1
AES encrypt	76,0	63,9
AES decrypt	75,3	49,9

Rich Normal World. Normal world applications are developed as regular Android applications. On the prototype, applications can be installed using the .apk files. On commercial platforms, an app store application can be provided. A Java library is provided to enable communication with secure World services. To test the performance impact of the secure world on the commodity OS, two tests have been run.

First, in order to test the overhead caused by activating the TrustZone security extensions, Android benchmarking tools have been used to compare the performance of the platform with and without TrustZone enabled. The benchmarking tools used are Antutu¹ and Geekbench². These tests were performed

¹ antutu.com/en/index.shtml.

² geekbench.com.

with the same Android version. The only difference on the platform was the activation of the TrustZone devices in one case. The results for these benchmarks can be found in Figs. 3 and 4. It is clear that activating TrustZone’s hardware security causes no noticeable overhead to the platform.

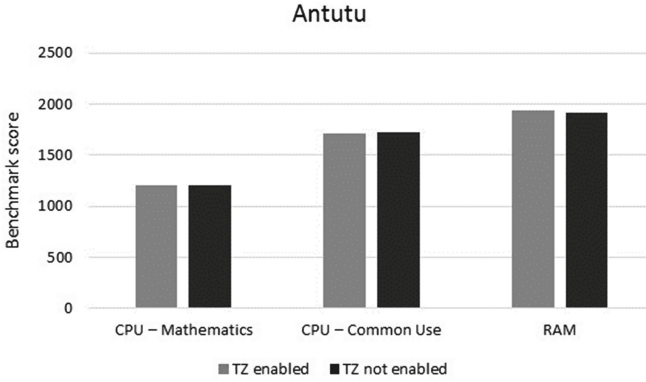


Fig. 3. Antutu benchmark results

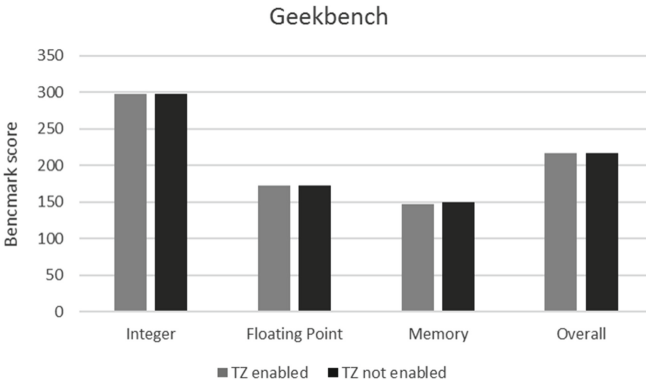


Fig. 4. Geekbench benchmark results

Second, the delay introduced by a world switch is measured using the Performance Monitor Unit. The counter is measured just before the world switch command is given and again when the activated world resumes execution. Table 2 shows the time it takes to switch worlds in either direction. The overhead introduced by the world switch is acceptable for most services. We notice a difference in the overhead depending on the direction of the world switch. This is due to how the switch is called in each world. In the normal world the smc is called directly so the timer can be started closely to the switch. In the secure world

there is more variety in the time until the switch is executed from when it is called due to how this call happens. The monitor requests the Genode kernel to make the switch but this is handled by the kernel's scheduler, so the workload of the machine has an influence on the overhead.

Table 2. Delay caused by world switch

Direction of world switch	Time in ms
From normal to secure world	0,069
From secure to normal world	0,411

7.2 Validation

This section shortly presents several use cases that illustrate the added value of the platform in the context of IoT and CPS. Future work will further explore the opportunities of the platform in these use cases.

Proprietary Local Data Processing: In the setting of Industrial Internet of Things, companies are developing services for the monitoring and management of industrial control systems. The algorithms used for this analysis are typically proprietary. Hence, to prevent competitors from obtaining these algorithms, they are currently mainly running on back-end systems. However, in some settings where analysis need to be performed on remote sites where Internet connectivity is not available or always guaranteed, the algorithms need to run in the field. Hence, a form of hardware-based confidentiality of the algorithm running in a secure environment is desirable, while allowing less trusted code on the platform to gather the data for the analysis and present the results to the user.

Leasing Machines: In the context of sustainable development and more durable customer relations, some companies are moving away from selling products (e.g. light bulbs, trains) and are selling services instead (e.g. lumination, locomotion). Companies are triggered to design products that have a longer lifespan and are easier to maintain. This can also lead to new business models in which a single device can be used by multiple customers. Hence, a set of trustworthy software components running on the device is required to handle these new business models and monitor the usage of the device. Less trusted software components could be loaded on the device to provide the interface and optionally other less critical services to the users.

Onboard Computer: In the automotive sector an increasing number of cars have an onboard computer. They have multiple internal networks and external networking capabilities. Some of the services running on these onboard computers such as park assist and engine start are critical, other services such as maps and infotainment are less critical. Further, in the transportation industry, the driving patterns need to be carefully monitored. Hence, a platform that supports

running security critical services while allowing an extensible software platform for non-critical software would provide great added value.

Home Care: In the health care sector, research and industry are intensively focusing on home care to manage the cost of the health care system. These systems allow the patient to be remotely monitored at their home. The medical gateway connects the monitoring devices to the Internet. The gateway can run critical medical services such as fall detection and heart-rate monitoring. However, an increasing number of health apps is being developed for consumers. Hence, a gateway on which these consumer apps could run next to more critical medical services can provide interesting opportunities.

7.3 Discussion

Extensions. The current setup has secure services running in a master-slave setting which limits the functionality of our platform. An interesting addition would be to have services activate periodically independent of the normal world. An extension which would allow this behaviour is to make use of the TrustZone watchdog timer. This is a timer exclusive to the secure world and which can be used to activate the secure world after a certain time regardless of the normal world's state. This extension would allow many new use cases to be serviced by our platform and is high on our priority list.

Another matter is installing new applications or updating existing software in the secure world. Currently, Genode does not allow on-the-fly installation of new services or the updating of existing software. When a software update needs to happen, the entire boot image needs to be rebuilt with the new code and settings and then redeployed. The creators of Genode, however, are striving towards supporting this functionality in later versions of the Genode OS. Once this has been realized, a secure service update and deployment scheme can be added to the platform.

Finally there is the matter of secure services accessing hardware resources. Since the secure world has access to all peripherals, the main issue herein lies with the security. Accessing peripherals through the normal world is a security vulnerability so the secure world needs drivers of its own to access these. Adding drivers increases Genode's TCB and thus increases the possibility of bugs and exploits.

Threat Model. Several TrustZone-based platforms have already been successfully attacked (e.g. Huawei HiSilicon TEE [15] and Qualcomm's Secure Execution Environment (QSEE) [8]). The attackers were able to extract security critical data from the secure environment. The attackers managed to control the Normal World OS so they could forge malicious commands for the Secure World. They then exploited vulnerabilities in the available commands to write arbitrary data to memory locations in the Secure World. Our platform provides a simple and strict communication API between both worlds. Further, an important aspect is the absence of bugs in the secure world. The secure world minimizes the trusted computing base of the secure services. For the current prototype, the secure

world software consists of approximately 24,000 lines of code. This suggests that techniques to verify the absence of specific vulnerabilities [6] such as buffer overflows could be used on the secure world code. To ensure that Genode correctly implements the isolation between the different secure services, techniques that verify that software is consistent with its specification could be used. For instance, the seL4 microkernel, consisting of 8,700 lines of C code and 600 lines of assembler, has been formally proven to be consistent with its specification and free from programmer-induced implementation errors [7]. The proof is constructed and checked in Isabelle/HOL [10], a proof assistant for higher-order logic.

8 Conclusion

This paper presents the design, implementation and evaluation of a platform which enables the development of secure rich CPS devices and gateways. This solution consists of a functional micro OS that provides a TEE for certain services. Confidentiality and integrity of the application code is offered in the isolated execution environment. These are achieved using ARM TrustZone and the Genode microkernel. A prototype of the platform is developed using the Freescale SABRElite i.MX6 development board. The evaluation of this prototype shows that it achieves a good level of security at an in general small performance cost.

References

1. Dolev, D., Yao, A.C.: On the security of public key protocols. *Trans. Inf. Theory* **29**(2), 198–208 (1983)
2. Feske, N.: Genode Operating System Framework 15.05
3. Freescale Semiconductor Inc.: i.MX6 Processor Reference Manual (2013)
4. Ge, X., Vijayakumar, H., Jaeger, T.: SPROBES: enforcing kernel code integrity on the trustzone architecture. In: *Proceedings of the Mobile Security Technologies 2014 Workshop* (2014)
5. Hofmann, O.S., Kim, S., Dunn, A.M., Lee, M.Z., Witchel, E.: InkTag: secure applications on an untrusted operating system. *ASPLOS* **2013**, 253–264 (2013)
6. Jacobs, B., Smans, J., Piessens, F.: A quick tour of the verifast program verifier. In: Ueda, K. (ed.) *APLAS 2010*. LNCS, vol. 6461, pp. 304–311. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17164-2_21
7. Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., Winwood, S.: seL4: formal verification of an OS kernel. In: *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, SOSP 2009*, pp. 207–220. ACM, New York (2009)
8. Laginimaineb: Extracting Qualcomm’s Keymaster Keys (2016). <http://bits-please.blogspot.be/2016/06/extracting-qualcomms-keymaster-keys.html>
9. Mayer, C.P.: Security and privacy challenges in the internet of things. *Electron. Commun. Eur. Assoc. Softw. Sci. Technol. ECEASST* **17**, 1–12 (2009)
10. Nipkow, T., Wenzel, M., Paulson, L.C.: Isabelle/HOL: Proof Assistant for Higher-order Logic, vol. 2283. Springer, Heidelberg (2002). <https://doi.org/10.1007/3-540-45949-9>

11. Noorman, J., Preneel, B., Agten, P., Daniels, W., Strackx, R., Huygens, C., Piessens, F., Van Herrewege, A., Verbauwhede, I.: Sancus: low-cost trustworthy extensible networked devices with a zero-software trusted computing base. In: 22nd USENIX Security (2013). K U Leuven
12. Santos, N., Raj, H., Saroiu, S., Wolman, A.: Using ARM TrustZone to Build a Trusted Language Runtime for Mobile Applications (i)
13. Santos, N., Raj, H., Saroiu, S., Wolman, A.: Trusted language runtime (TLR): enabling trusted applications on smartphones. In: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications (HotMobile), pp. 21–26 (2011)
14. Freescale Semiconductor: Secure Boot on i.MX50, i.MX53, and i.MX 6 Series using HABv4, pp. 1–22 (2012)
15. Shen, D.: Exploiting Trustzone on Android. Black Hat (2015)
16. Strackx, R., Noorman, J., Verbauwhede, I., Preneel, B., Piessens, F.: Protected software module architectures. In: Reimer, H., Pohlmann, N., Schneider, W. (eds.) ISSE 2013 Securing Electronic Business Processes, pp. 241–251. Springer, Wiesbaden (2013). https://doi.org/10.1007/978-3-658-03371-2_21
17. Zha, Z., Li, M., Zang, W., Yu, M. and Chen, S.: AppGuard: a hardware virtualization based approach on protecting user applications from untrusted commodity operating system. In: 2015 International Conference on Computing, Networking and Communications, ICNC 2015, pp. 685–689 (2015)

Introducing Usage Control in MQTT

Antonio La Marra¹, Fabio Martinelli¹, Paolo Mori¹, Athanasios Rizos^{1,2},
and Andrea Saracino¹(✉)

¹ Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy
{antonio.lamarra,fabio.martinelli,paolo.mori,athanasios.rizos,
andrea.saracino}@iit.cnr.it

² Department of Computer Science, University of Pisa, Pisa, Italy

Abstract. MQTT is a widely-used general purpose IoT application layer protocol, usable in both constrained and powerful devices, which coordinates data exchanges through a publish/subscribe approach. In this paper we propose a methodology to increase the security of the MQTT protocol, by including Usage Control in its operative workflow. The inclusion of Usage Control enables a fine-grained dynamic control of the rights of subscribers to access data and data-streams over time, by monitoring mutable attributes related to the subscriber, the environment or data itself. We will present the architecture and workflow of MQTT enhanced through Usage Control, also presenting a real implementation on Raspberry Pi 3 for performance evaluation.

1 Introduction

Over the last years, Internet of Things (IoT) devices have become more and more pervasive to our daily life. IoT devices could be very different, since they typically have different types of hardware, depending on the provided functionalities, and software applications to manage them. Hence, in order to have a unique application which eases the control of all the smart devices owned by the same user, a necessity has arisen to be able to easily communicate with a set of distinct IoT devices. To this aim, several protocols have been proposed in the scientific literature, and among them, MQTT, which is recently standardized by OASIS, is one of the most widely used [1]. Since MQTT is based on HTTP functionalities, most of the MQTT security solutions seem to be either application specific, or just leveraging TLS/SSL protocols [1]. Although the effort concerning security of MQTT protocol is rising, two main obstacles occur. The first one is that, although the protocol has the ability to deal with various Publishers and Subscribers, since they use different platforms, it is difficult to create and enforce generic security policies. The second problem is that the current efforts are mainly directed to standard message communication security. Still no efforts

This work has been partially funded by EU Funded projects H2020 C3ISP, GA #700294, H2020 NeCS, GA #675320 and EIT Digital HII on Trusted Cloud Management.

have been done in the direction of supporting policy enforcement at Broker level, nor it has been considered the possibility of dynamically revoking subscriptions.

In this paper, we propose the enhancement of the security of the MQTT protocol by adding Usage Control (UCON) in the MQTT architecture and workflow. UCON is an extension of traditional access control which enforces continuity of access decision, by evaluating policies based on mutable attributes, i.e. attributes changing over time [7]. By adding Usage Control in MQTT, we aim at enforcing dynamically fine grained policies, which do not only consider the identity of the Subscriber as a parameter for granting access to data, but also dynamic attributes such as Subscriber reputation, data reliability, or environmental conditions of a specific application. After surveying the main IoT application protocols, and motivating the choice of focusing on MQTT, this work will discuss the architecture and the workflow of the MQTT - UCON integration. The proposed framework is designed to be general, easy to integrate in the Broker component, remaining oblivious to both Publishers and Subscribers. The addition of UCON in fact, does not modify the MQTT protocol, enforcing the policies independently from the implementation of Publisher and Subscriber, which allows the proposed solution to be compatible with any off-the-shelf MQTT Publisher/Subscriber application. Furthermore, we demonstrate the viability of the approach by presenting a real implementation of the framework on both general purpose and performance constrained devices, measuring the performance on two Raspberry Pi 3 model B¹ that are used respectively as Broker and Subscriber.

The rest of the paper is organized as follows: In Sect. 2, a comparison between the main IoT application protocols is reported, detailing afterward the MQTT protocol and motivating our choice to focus on it. Furthermore, some background information about usage control are reported. Section 3 describes the integration of UCON and MQTT detailing the architecture and the operative workflow. Section 4 details the results of the performance analysis. In Sect. 5 are reported a set of related works about security in MQTT and application of UCON in IoT. Finally, Sect. 6 concludes by proposing future directions which stem from this preliminary work.

2 IoT Protocols and MQTT

The most known application layer protocols in IoT are CoAP, MQTT, XMPP, HTTP, AMQP and WebSocket. CoAP is more resource-friendly than MQTT [5] but in terms of Message Oriented Approach (MOA), MQTT stands out. All the protocols mentioned above use TCP as transport layer. Only CoAP uses UDP. The same happens as for the security layer. All protocols use TLS/SSL except from CoAP that uses DTLS. In fact, CoAP targets to very constrained environments. Furthermore, according to [6], MQTT provides the smallest header size of two bytes, although it is based on TCP. Moreover, it provides three levels of QoS which puts this protocol in the first place in terms of QoS, even though it needs extra load in the network for message retransmission. On the other hand,

¹ <https://www.raspberrypi.org/products/raspberrypi-3-model-b/>.

XMPP requires processing and storing XML data, which necessitates memory space that is too large for most IoT devices. In addition, HTTP performs better in non constrained environments when PC, Laptop and Servers are used. It is generally not applicable in IoT devices due to its high overhead. AMQP [9], is more suitable for server-to-server communication than device-to-device communication. Websocket is neither a request/response nor a Publish/Subscribe protocol. In Websocket, a client initializes a handshake with a server to establish a Websocket session. The handshake process is intended to be compatible with HTTP-based server-side software so that a single port can be used by both HTTP and Websocket clients [4]. According to [12], MQTT messages experience lower delays than CoAP for lower packet loss and vice versa. When the message size is small the loss rate is equal. AllJoyn [13], is a full stack of protocols intended for IoT. Though quite popular, the main disadvantage of AllJoyn is that the application protocol cannot be separated from the rest of the protocol stack. As a synopsis to the basis, reader can also consult Table 1. This comparison gives the details about the existence of Quality of Service (QoS). Moreover, it refers to the communication pattern, which in the case of MQTT is the Publish/Subscribe. The most significant column is the third. In this column, we identified that MQTT is more general purpose.

Table 1. Application layer protocol comparison

Protocol	QoS	Communication pattern	Target devices
CoAP	Yes	Req/Resp	Very constrained
MQTT	Yes	Pub/Sub	Generic, small header
XMPP	No	Req/Resp Pub/Sub	High memory consumption
HTTP	No	Req/Resp	High performance
AMQP	Yes	Pub/Sub	Ser-2-Ser communication
Web Socket	No	Client/Server Pub/Sub	Needs less power than HTTP still needs high power
AllJoyn	No	Client/Server Pub/Sub	High computational power

MQTT is an open pub/sub protocol where the system complexities reside on the Brokers side. MQTT does not specify any routing or networking techniques; it assumes that the underlying network provides a point-to-point, session-oriented, auto-segmenting data transport service with in-order delivery (such as TCP/IP) and employs this service for the exchange of messages. MQTT is a topic-based Publish/Subscribe protocol that uses character strings to provide

support of hierarchical topics. This also gives also the opportunity to the subscription to multiple topics. MQTT supports basic end-to-end Quality of Service (QoS) [3]. Depending on how reliably messages should be delivered to their receivers, MQTT provides three QoS levels. QoS level 0 is the simplest one: it offers a best -effort delivery service, in which messages are delivered either once or not at all to their destination. No retransmission or acknowledgment is defined. QoS level 1 provides a more reliable transport: messages with QoS level 1 are retransmitted until they are acknowledged by the receivers. This means that QoS level 1 messages may arrive multiple times at the destination because of the retransmissions. The highest QoS level, QoS level 2, ensures not only the reception of the messages, but also that they are delivered only once.

3 Introducing UCON in MQTT

In this section we present the proposed architecture, presenting first the model, then the operative workflow and the performed implementation. As previously mentioned, MQTT protocol is based on the Publish/Subscribe model, thus the entities participating to the protocol can act either as *Publishers* or as *Subscribers*. Publishers could be sensors or other devices which collect and provide specific data, when available, periodically or even as a stream. Subscribers are instead entities that register to the Broker to receive, when available, specific data or set of information grouped under a *Topic*. The *Broker* acts as middleware and coordinator, managing the subscription requests and dispatching data to Subscribers, when available by prosumers.

The MQTT protocol supports ID and password-based authentication for both Publishers and Subscribers. The enforcement is performed on Broker's side, which keeps track of the ID and authentication password of authorized Publishers and Subscribers. However, we argue that this authentication model is too simplistic and coarse grained, making impossible to check the right to access information over time. In fact, once a Subscriber has been authorized, the subscription remains valid until the Subscriber explicitly invokes an `unsubscribe` for the topic(s) it was registered for. The same goes for Publishers which keeps the right to publish continuously or on demand, till they have valid credentials. In real applications, several features might imply a condition for a subscription to decay, or for a publication to be denied. Detected Publisher malfunction or corruption, conditions on time spans in which a subscription should be allowed, and Subscriber reputation, are just few examples of aspects on which a more complex policy should be enforced. To be able to enforce policies with similar conditions to the aforementioned ones, and to have the possibility of revoking a subscription, usage control has been added to the MQTT logical architecture. In Fig. 1, we depict the logical architecture of the proposed framework.

As shown, the UCS is physically integrated in the Broker Device, i.e. the physical machine that is hosting the Broker software, which enables the MQTT protocol. It is worth noting that we consider in this example three abstract PIPs, which are conceptually grouping the PIPs reading attributes related to the subject (PIP_S), to the resource (PIP_R) and to the environment (PIP_E). The PEP

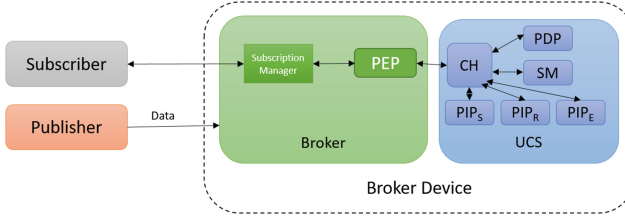


Fig. 1. UCON implementation in MQTT.

is (partially) embedded in the broker, to dynamically control the subscription events. In particular, the PEP will intercept the subscription events and interact directly with the Broker subscription manager, deleting and inserting the entries for Subscribers from the list of authorized ones, according on the UCS decision. In such a way, the PEP ensures that no Subscribers can register by avoiding the enforcement of the usage control policy. Since the PEP is embedded in the Broker, the proposed architecture remains compatible with any implementation of MQTT Subscribers. The only requirement is that the Subscriber is configured to access with username and password, otherwise the connection will be refused by the broker. In Fig. 2, we report the envisioned workflow. For the sake of simplicity, we will consider a simple system made out of a Broker and a single Publisher and Subscriber. The workflow is initiated by a subscription request from the Subscriber to the Broker. This request is intercepted by the PEP, which interprets it, so as to take the credentials of the Subscriber that are needed in order to create and send the request to the UCS for evaluation. Hence the PEP invokes the `TryAccess` sending to the CH request and policy. The request is eventually filled by attributes retrieved through the PIP, then is sent, together with the policy, to the PDP for evaluation, which should return a Permit or Deny decision. In case of Deny, the subscription request is dropped and the Subscriber will be notified, as if a wrong username/password has been inserted. In case of Permit, the Session Manager (SM) creates the session and sends its ID to the PEP (via the CH) which is informed about this decision and performs the `StartAccess`. Supposing a permit decision has been received, the Broker informs the Subscriber about the successful subscription and starts to send data related to the topic when available, eventually stimulating Publishers in an idle state. To illustrate the `revoke` workflow, we suppose that one of the attributes relevant for the Subscriber policy change its value (`OnAttributeUpdate`). This causes the PIP to send this new attribute to the CH that forwards it to the PDP for reevaluation. Supposing that the value of this attribute leads to a conclusion that this session must be revoked (Deny decision), the CH invokes the `RevokeAccess` on the PEP, also informing the Subscriber that the access is no longer granted (`RevokeAccess`). The termination of the access could happen also if the Subscriber is no longer interested to the data, invoking the `Unsubscribe`. The unsubscribe triggers the PEP to send an `EndAccess` to the CH. The latter informs the PIP to take the last value of the attribute (`PostAttributeUpdate`).

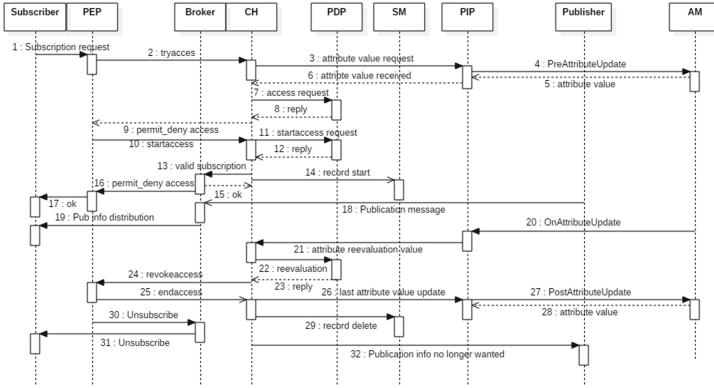


Fig. 2. Full workflow sequence diagram.

Also the UCS informs the Broker that the Subscriber is no longer subscribed and forces the unsubscription. Moreover, the SM is also informed that this session is over so that the record should be archived or deleted. Finally, if this Subscriber is assumed to be the only one that was interested to the Publisher, the Broker informs him to stop data publication due to fact that there is no more any interest from any Subscriber.

4 Results

To demonstrate the viability of the proposed approach, the overhead introduced by usage control has been measured in a simulated and in a real environment. The framework has been tested in two different environments: the first one is a virtual machine installing Ubuntu 16.04 64-bit, equipped with an Intel i7-6700HQ with 8 cores enabled, 8 GB DDR4 RAM running in 2133 MHz, the second one is a Raspberry Pi 3. In Fig. 3, there is reported the performance variation at the increase of the number of attributes used in the policy. As shown, the timing behavior is almost linear to the amount of attributes, which is expected, due to the longer time needed to collect a larger number of attributes and for the evaluation performed by the PDP. However, in the real case, even considering 40 attributes, the timings are still acceptable for most of applications. Moreover, it is worth noting that policies with a large number of attributes such as 40 are quite unusual [10]. As expected, the low computational power of the Raspberry alongside the existence of a real network among the MQTT components, justify the longer timings than in the simulated environment. Considering the subscription time, we see that there is some overhead caused by UCS. This is not be considered as a constraint because, since the Broker provides a buffer, we can still send all the published messages between the time of the request and the actual acceptance of the Subscriber. This causes no packet loss to the Subscriber and high QoS. Furthermore, the most significant time is the one of

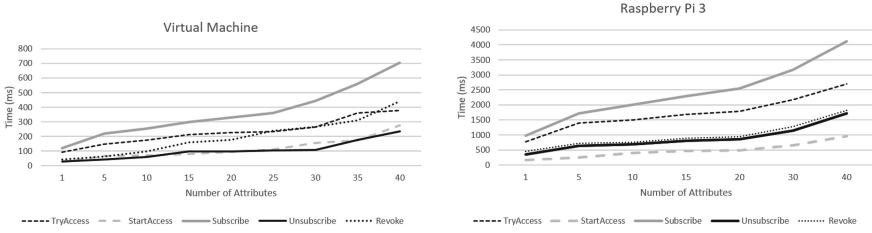


Fig. 3. Timings on the simulated and real testbed.

the revocation. This time is in fact the actual time in which the policy is violated and should be minimized. As shown, this time is equivalent to 216 ms in the real use case and 27 ms in the Virtual Machine, considering a policy with a single attribute. For several applications, this time can be considered as negligible. As shown, the time between a non valid value is taken and revocation of the access is very small. Finally, it is worth mentioning that in the ongoing phase, i.e. after a successful `StartAccess`, no delay is introduced by UCON while delivering messages to the Subscribers independently also of the number of attributes.

5 Related Work

Although there exist applications of UCON in GRID [11] and Cloud [2] systems, there is only one targeting on IoT [10]. In this paper the authors present a version of Usage Control called UCIoT that aims to bring the UCON on IoT architectures. Their work mainly focuses on an implementation of UCIoT in a smart home environment. They present specific policies alongside experiments on testbed. They do not, though, state how this framework (UCIoT) can be applied to the several application protocols. Their implementation in a P2P environment does not state how the UCON framework deals with each protocol. Our solution addresses this lack of addressing these protocols and how UCON can work alongside them. The authors of [8] propose a solution to securing Smart Maintenance Services. Their goal is to proactively predict and optimize the Maintenance, Repair and Operations (MRO) processes carried out by a device maintainer for industrial devices deployed at the customer. They focus on the MQTT routing information asset and they define two elementary security goals regarding the client authentication. Their solution is based on Transport Layer Security (TLS) which is already a basic feature of the protocol. They proposed on how to use it more efficient as a hardware element. Although they claim that the performance impact is not significant, the adoption of an extra hardware component might be critical in the constrained environment of IoT.

6 Conclusion

In this paper we have presented a first preliminary effort to increase the security of the MQTT protocol, by enabling the dynamic enforcement of Usage

Control policies. We have presented a general methodology which allows to integrate UCON in a seamless way, without requiring protocol modifications. A real implementation has been presented, with performance evaluation to demonstrate the viability of this approach. As future work we plan to test the presented framework on a larger testbed with a larger number of Publishers and Subscribers for the definition and enforcement of more complex policies, with a possible evaluation in a real applicative setting. Furthermore, we point out that the applied methodology can be easily extended to other IoT application protocols, where the benefits of integration are worth to be investigated in future works.

References

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of Things: a survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutorials* **17**(4), 2347–2376 (2015)
2. Carniani, E., D’Arenzo, D., Lazouski, A., Martinelli, F., Mori, P.: Usage control on cloud systems. *Future Gener. Comput. Syst.* **63**(C), 37–55 (2016)
3. Chen, D., Varshney, P.K.: QoS support in wireless sensor networks: a survey. In: *International Conference on Wireless Networks*, vol. 233, pp. 1–7 (2004)
4. Colitti, W., Steenhaut, K., De Caro, N., Buta, B., Dobrota, V.: Evaluation of constrained application protocol for wireless sensor networks. In: *2011 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN)*, pp. 1–6, October 2011
5. Fysarakis, K., Askoxylakis, I., Soultatos, O., Papaefstathiou, I., Manifavas, C., Katos, V.: Which IoT protocol? Comparing standardized approaches over a common M2M application. In: *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7. IEEE (2016)
6. Karagiannis, V., Chatzimisios, P., Vázquez-Gallego, F., Alonso-Zrate, J.: A survey on application layer protocols for the Internet of Things. *Trans. IoT Cloud Comput.* **1**(1), 11–17 (2015)
7. Lazouski, A., Martinelli, F., Mori, P.: Usage control in computer security: a survey. *Comput. Sci. Rev.* **4**(2), 81–99 (2010)
8. Lesjak, C., Hein, D., Hofmann, M., Maritsch, M., Aldrian, A., Priller, P., Ebner, T., Rupprechter, T., Pregartner, G.: Securing smart maintenance services: hardware-security and TLS for MQTT. In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pp. 1243–1250, July 2015
9. Luzuriaga, J.E., Perez, M., Boronat, P., Cano, J.C., Calafate, C., Manzoni, P.: A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. In: *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 931–936, January 2015
10. La Marra, A., Martinelli, F., Mori, P., Saracino, A.: Implementing usage control in Internet of Things: a smart home use case. In: *2017 IEEE Trustcom/BigDataSE/ICCESS*, Sydney, Australia, 1–4 August 2017, pp. 1056–1063 (2017)
11. Martinelli, F., Mori, P.: On usage control for grid systems. *Future Gener. Comput. Syst.* **26**(7), 1032–1042 (2010)

12. Thangavel, D., Ma, X., Valera, A., Tan, H.X., Tan, C.K.Y.: Performance evaluation of MQTT and CoAP via a common middleware. In: 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–6. IEEE (2014)
13. Villari, M., Celesti, A., Fazio, M., Puliafito, A.: Alljoyn lambda: an architecture for the management of smart environments in IoT. In: 2014 International Conference on Smart Computing Workshops, pp. 9–14, November 2014

Threats, Vulnerabilities and Risks

Towards Security Threats that Matter

Katja Tuma¹(✉), Riccardo Scandariato¹, Mathias Widman²,
and Christian Sandberg³

¹ University of Gothenburg, Gothenburg, Sweden
`katja.tuma@cse.gu.se`

² Wireless Car, Gothenburg, Sweden

³ Volvo AB, Gothenburg, Sweden

Abstract. Architectural threat analysis is a pillar of security by design and is routinely performed in companies. STRIDE is a well-known technique that is predominantly used to this aim. This technique aims towards maximizing completeness of discovered threats and leads to discovering a large number of threats. Many of them are eventually ranked with the lowest importance during the prioritization process, which takes place *after* the threat elicitation. While low-priority threats are often ignored later on, the analyst has spent significant time in eliciting them, which is highly inefficient. Experience in large companies shows that there is a shortage of security experts, which have limited time when analyzing architectural designs. Therefore, there is a need for a more efficient use of the allocated resources. This paper attempts to mitigate the problem by introducing a novel approach consisting of a risk-first, end-to-end asset analysis. Our approach enriches the architectural model used during the threat analysis, with a particular focus on representing security assumptions and constraints about the solution space. This richer set of information is leveraged during the architectural threat analysis in order to apply the necessary abstractions, which result in a lower number of significant threats. We illustrate our approach by applying it on an architecture originating from the automotive industry.

Keywords: Architectural threat analysis · Security assets · STRIDE

1 Introduction

In an ever more complex Cyber-Physical System (CPS) domain, security and trust management are becoming burdensome for many organizations. New software products and frameworks are intended to support functionalities that handle privacy and security of sensitive data. Furthermore, the longevity of a CPS product is typically high (e.g., in the automotive, 25 years), which makes building a secure solution a substantial challenge. Security by design requires addressing security-related issues throughout the entire software development life-cycle. This paper focuses on the early stage of conceptualization of a software system, i.e., the architectural design. Planning for security in early design phases

helps designers to steer the product development in a direction where threat mitigations are possible.

In particular, threat analysis is a method that strives towards validating the software architecture and discovering potential design weaknesses. This validation technique is an essential pillar of software security, together with other code-level verification techniques like static analysis and security testing [17]. For instance, Microsoft’s STRIDE is a well-known and used technique to perform architectural threat analysis [20]. STRIDE and similar techniques (like LIND-DUN [25]) follow the so-called software-centric approach. Such techniques center the analysis around a model of the system that resembles a graph and represents the software components (both computation and storage nodes) and the information exchanged between them (edges). From a syntactical perspective, Data Flow Diagrams (or DFD) are often used to represent such models. According to STRIDE the analysis proceeds by exploring the diagram and discovering several potential threats at each location. On one hand, this way of performing a security assessment has the benefit of being systematic. On the other hand, the analysis is prone to being repetitive and very time consuming. Empirical evidence shows that with only a handful of software components, the analysis can result in the discovery of 50 to 60 security threats, which means a scale factor of 10 [18]. This is known as the ‘threat explosion’ problem.

The experience of our industrial partners is that, trading systematicity for a timely discovery of most important threats is advantageous. As resources are scarce and time is limited, systematicity is considered an obstacle if it leads to ‘wasting’ time with security threats that are deemed as not important later on. We also learned that in the early design phase, stakeholders reason about security with a close eye on system assets. Rather than focusing the analysis on the software assets (e.g., software components and data stores), analysts observe information assets and how they move through the system. They do that by analyzing end-to-end usage scenarios which involve a certain information asset and trace the software components that are involved with exchanging, using and storing that particular asset. At each encountered software component, they reason about the potential threats to the asset.

In this paper we synthesize the lessons learned while analyzing the threats in an architecture of a connected vehicle and suggest a novel way of approaching threat analysis. We propose an architectural view for security that is based on DFDs, extended with end-to-end flows representing the information assets in the system. In our enriched model, assets are also annotated with their importance and with security objectives associated with them (e.g., confidentiality). The extended model also includes additional information that is routinely used during the threat analysis process, namely, security assumptions. The extended notation is used to guide the threat analysis and reduce the amount of ‘uninteresting’ threats that are found. For instance, if an end-to-end flow refers to an information asset that needs to stay confidential, it would be better to focus on disclosure threats (which directly impact confidentiality) rather than on denial-of-service threats (which impact availability instead).

In order to illustrate our approach, the first author applied it on the architecture provided in the context of the HoliSec project [4] on security of connected vehicles. The results of the analysis have been submitted to a domain expert with extensive security background (the third author). Our discussions concluded that our approach indeed led to the identification of valid threats, likely to have the most impact to the organization in reality. Clearly, the obtained results serve as a proof of concept and are a stepping stone for future work.

The remainder of this paper is structured as follows. Section 2 describes the running example. Section 3 presents the extended notation and the needs of the analyst, Sect. 4 presents the guidelines for abstracting the architectural model (DFD) applied to the running example and Sect. 5 identifies the related work. Finally, Sect. 6 includes a discussion and future work, followed by concluding remarks, presented in Sect. 7.

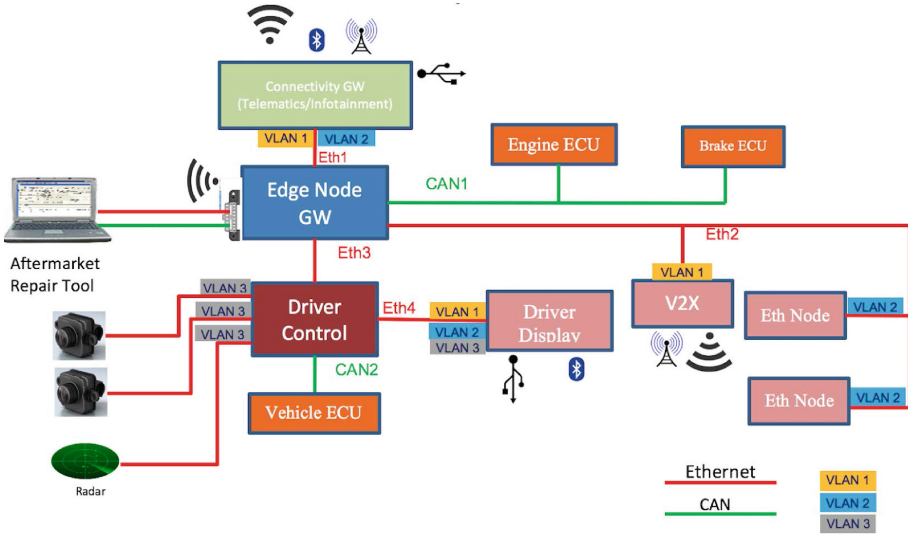
2 Running Example

In this section we describe the architecture of a vehicle as shown in Fig. 1. This example is used throughout the paper to illustrate the benefits of our proposed approach. Due to non-disclosure concerns, the example is realistic but does not correspond to an architecture of an existing product on the market.

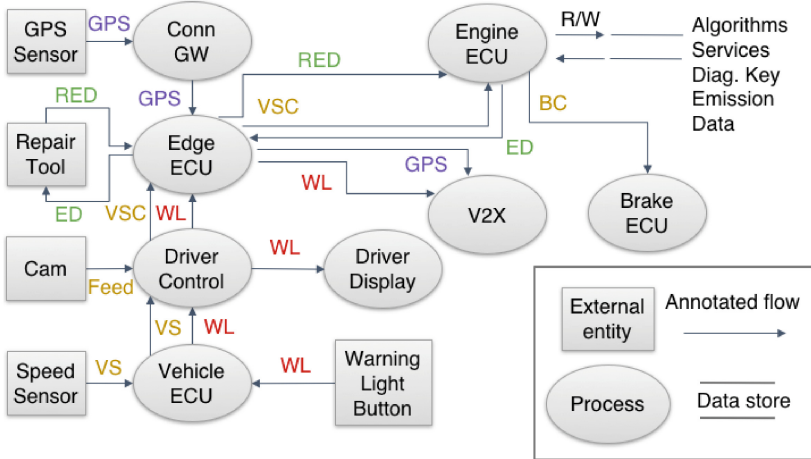
Modern vehicles are highly complex systems, comprised of hundreds of different components called Electronic Control Units (ECUs), which are responsible for one or more particular features of the vehicle. Individual ECUs are connected to a Controller Area Network (CAN) bus, which is currently the most used in-vehicle communication protocol and also a very common target of attack [26].

As depicted in Fig. 1a, the architecture is composed of several ECUs, sensors and actuators exchanging data between each other following a specific communication protocol. The communication between individual components is further specified with a communication matrix (not shown here) of signals, source and receiver components, networks used and type of communication (e.g., broadcast or unicast). For instance, the warning light signals are broadcast on networks CAN 2, Eth 2, Eth 3 and Eth 4. The architecture in Fig. 1a supports a number of functional scenarios, which are described below. Figure 1b presents a DFD, derived from the architectural information and the assets described in the scenarios. Notice that functional elements represent *processes*, while *data stores* represent the places where information is stored for later retrieval. Everything that is outside of the system (e.g., 3rd party systems) is modeled by means of *external entities*. The arrows represent the exchange of information.

Scenario 1: Set-up diagnostics connection and read emission data. A logging functionality collects information about the vehicle over time, such as the emissions data and the GPS position. In order for the data to be collected, the Repair Tool sends the emission data request signal (RED) via the Edge ECU to Engine ECU over the OBD network. The Engine ECU then sends the emission data response signal (ED), including the requested information, back to the external interactor.



(a) Architecture of in-vehicle communication.



(b) DFD of the architecture.

Fig. 1. The running example

Scenario 2: Extended vehicle warning. Vehicle to X (V2X) communication allows the exchange of information between the vehicle and the road infrastructure or other vehicles. If the warning light button is active, the vehicle forwards the warning light status (WL) from the Vehicle ECU to the Driver Control. The latter sends it over the Eth 4 network to the Driver Display in order to alert the driver. The warning light status is then sent over the Eth 1 network to the Edge ECU, which, in turn, collects the current GPS position from the

Connectivity Gateway. The information (both WL status and GPS position) is sent to the Edge ECU via the Driver Control ECU on networks CAN 2 and Eth 3. Finally, both the GPS location and the warning light status are broadcast via the V2X ECU.

Scenario 3: Using city traffic collision prevention to brake for pedestrians in the trajectory of the vehicle. This scenario describes the situation where city traffic collision prevention is used to slow down or stop the vehicle if pedestrians appear in the vehicles' trajectory. The camera sensor sends live video feed to the Driver Control ECU, which analyses it for upcoming obstacles. The Driver Control ECU also receives information about the vehicle speed (VS) from the Vehicle ECU. If a possible collision is detected, the Driver Control ECU generates a vehicle speed change request (VSC) and sends it to the Engine ECU, which orderly sends a brake command (BC) to the Brake ECU.

3 An Extended DFD Notation

Security experts performing threat analysis are aware of the threat explosion problem and try to counter it by making abstractions. For instance, analysts often group seemingly homogeneous elements of the DFD with respect to the type of threats they are subject too. In STRIDE, this technique is called reduction [7]. To make any sort of abstractions possible, candidate elements need to be closely inspected in order to decide whether an abstraction will overall have a negative or a positive outcome. This decision should be an intelligent choice, supported by evidence in the architecture. In order to trace assets through the system, we propose to use asset-centric partial DFDs. Figure 2 shows a DFD for the *vehicle speed (VS)* asset. Notice that the DFD is a slice of the overall model presented in Fig. 1b.

Security objectives and priorities. Each data flow is marked with the asset that is being transported (also part of the standard DFD notation). We extend the notation by introducing clear markers of where an asset is generated (*asset source*) and where it is consumed (*asset target*). If assets are to be protected, they have to be analyzed from the source all the way to the target architectural element. Additionally, the asset is labeled with one or more *security objectives*: confidentiality (C), integrity (I), availability (A). Security objectives are used to focus the identification of important threats during threat analysis. Often, the analysis spans across several months with brief sessions every week or two. After a few sessions, it is easy to forget about the analysis constraints (e.g., 'focus on confidentiality only') if they are not clearly visible in the diagram. The asset is also ear-marked with a *priority* label that signifies the importance of the objective. We use the values of low (L), medium (M) and high (H). These do not express the importance of the asset as such, but rather the impact of a compromised asset objective. This information helps calibrate the depth of the analysis to come.

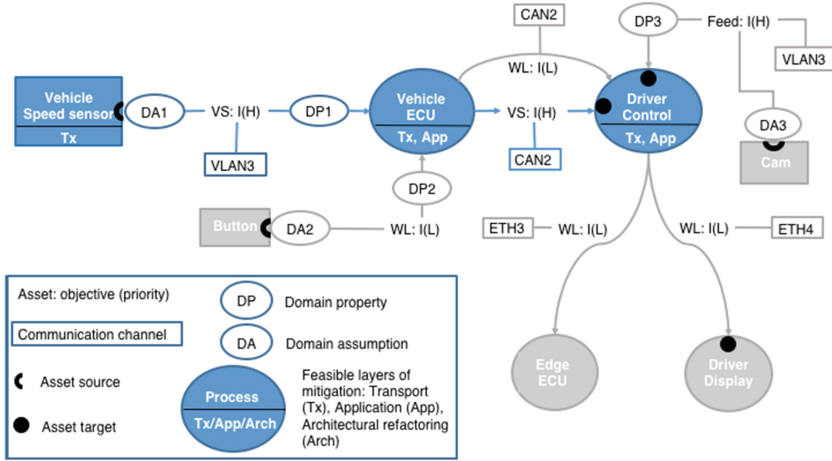


Fig. 2. The extended DFD notation for an end-to-end, asset-centric flow.

Security assumptions and properties (at flows). During threat analysis, assumptions are made in order to assess whether threats are feasible in the underlying architecture. For instance, the integrity of VS can be compromised if it is transported in clear and if the attacker has access to the transport medium. Together with the domain expert, the analyst may choose to make an assumption about an existing security mechanism in place that protects VS against tampering threats. When making assumptions about the system, the analysts need to be very careful not to make optimistic assumptions, which can lead to overlooked threats. A false assumption about the security of a GPS sensor can, for example, result in overlooking spoofing threats. On the other hand, not making any assumptions can make the analysis highly inefficient and result in the elicitation of irrelevant threats. Today, assumptions are sometimes still documented separately in an informal way. Considering that they are easily forgettable, they must be made visible in the model, right where they are needed.

In this paper, we distinguish between domain assumptions and domain properties, as described by Van Lamsweerde [24]. Domain properties are used to describe non disputable facts about the domain, whereas domain assumptions are statements about the domain that may or may not hold. For instance, “It is infeasible to encrypt the CAN bus” is a domain property. This is something that we can not change about the domain. On the other hand, “The CAN bus is not accessible to the attacker” is a typical domain assumption.

In Fig. 2 there is one assumption and one property on top of the flow going from vehicle speed sensor to Vehicle ECU. The domain assumption DA1 is the following: “The vehicle speed sensor is a Commercial-Off-The-Shelf product and is working securely.” The domain property DP1 placed on the same flow is the following: “There is a feasible mitigation solution on the transport layer for the flow between the vehicle speed sensor and the Vehicle ECU.” The assumption

DA1 and property DP1 are later on used to argue about the security of assets on the flow, which is discussed when abstracting the diagram. The limited layers of mitigation solutions are annotated within processes and external entities with capital letters.

Finally, there is one property that is so important (especially in embedded systems) that it gets its own annotation. We refer to the representation of the *communication channel* for each data flow. The communication channel explicitly shows which network the data flow and the corresponding asset belong to. A regular DFD notation does not include the topological behavior gathered in the communication matrix. Keeping that information visible is important, because most domain properties and assumptions that have to be made are about network and protocol capabilities.

Forward assumptions (at processes). Processes are ear-marked with this annotation, which is important in the perspective of simplifying the analysis process, as described in Sect. 4. In essence, we suggest that it is useful to explore the space of possible solutions which are realistic to implement in terms of mitigation mechanisms. For example, some ECUs include a Hardware Security Model (HSM), which provides transport layer encryption. This exploration needs to be done before the threat analysis starts. As the same threats start to appear more often (e.g., the assessment of the integrity of the VS is generating a lot of tampering threats along the asset flow), it is more efficient to turn a forward assumption into a domain property (i.e., mandate the adoption of a certain security mechanism, like turning on the encryption) and stop bothering about that asset all together. This kind of backtracking in the analysis process is supported by the extended notation.

Our work differentiates between threat mitigation solutions on different layers of abstraction: transport (Tx), application (App) and architectural (Arch) layer. For instance, on the transport layer, symmetric cryptography may be used to establish a secure communication protocol between one of the sensors connected to the vehicle and the receiving ECU. On the other hand, an application layer mitigation solution would include an application layer firewall that inspects packets traveling to and from the Repair Tool in a remote diagnostic scenario. On the architectural layer, security mitigation techniques include architectural re-factoring, where possible design decisions are required to modify the system architecture. Note that middle-ware solutions, such as message queues, are also grouped as architectural layer mitigation techniques.

In addition, while the focus of abstraction is on a single end-to-end flow, elements indirectly involved in the end-to-end flow are still represented in the diagram. The reason for including additional elements is because abstractions directly effect neighbor elements. Consider what happens if two processes are folded. One part of the end-to-end flow gets successfully abstracted. However, there might have been other flows between that were unaccounted for. Neglecting the neighboring elements can increase the risk of missing important threats during the analysis. We use a different color for such elements (gray) to stress this point.

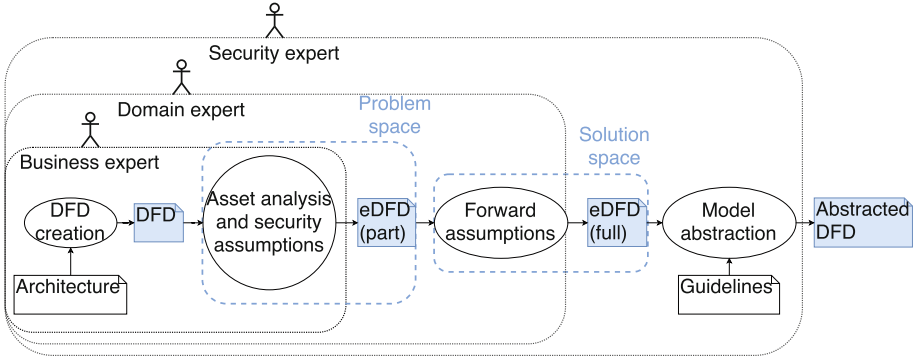


Fig. 3. Roles and responsibilities for extending the DFD using the proposed approach.

In summary, there are three important actors that participate in the process of obtaining the extended DFD. Figure 3 shows how a domain and a security expert work together with a business expert to gather the necessary information. The eDFD is built after having analyzed the problem and solution space. First, the domain expert defines an architectural model including security objectives, purpose and priority of main assets in the system. To that end, the business expert contributes with determining the priorities of assets, while the security expert helps define the objectives. The architectural model is comprised of a structural, behavioral and a topological view. This architecture is used as input to create a DFD. After the DFD defined, all three actors contribute to asset analysis, where they reason around the *problem space*. Asset analysis is performed in an iterative manner. Each asset is first identified, then the source and target components of the asset are located. Although the domain and security experts are mostly active in this phase, the business expert contributes to the discussion from a financial perspective. The asset is then traced in the DFD, where all components affiliated with the asset are marked accordingly. Asset analysis activities are repeated for all assets identified within user scenarios (behavioral view). At this point, the partially extended DFD includes security objectives, priorities and security assumptions at flows, put forward in Sect. 3. In order to complete the eDFD, forward assumptions about processes need to be made. When making forward assumptions about security mitigations, the actors discuss the so called *solution space*. To wit, domain and security experts limit possible mitigations in accordance with domain specific constraints and possibilities. Lastly, using the complete eDFD, the security expert makes use of the guidelines to abstract the DFD. Section 4 discusses the model abstraction activity and how it counters the threat explosion. Threat analysis begins after the DFD is abstracted.

4 Handling the Threat Explosion

In this section we discuss how our approach tackles the previously mentioned problem of the ‘threat explosion’, i.e., when too many (often irrelevant) threats are found when STRIDE is applied to a medium-to-large DFD. With the support of the running example, we illustrate how the abstraction is performed before the threat analysis and how the effort reduction is supported during the analysis.

4.1 Abstraction Before Threat Analysis

The first approach towards solving the problem is to reduce the number of DFD elements (before the analysis starts) by means of heuristics. In such a manner the model is simplified and consequently, the analysis produces less unimportant threats. We have defined two initial guidelines for flow bundling and process folding. To this aim, the extended notation introduced previously plays an important role. The guidelines were obtained through iterative sound-boarding with industrial partners while working on the architecture from Fig. 1. The reader should note that the guidelines are not meant to be a complete set of criteria, but rather the result of our initial observations. Having said that, we defined two different sets of guidelines for components with either critical or non-critical assets. For critical components, a more strict set of criteria is used, while for non-critical components the criteria are somewhat loosened. In this way, the abstraction is done in accordance with the “heat” of the system (obtained from the asset analysis) in a each region.

Bundling data flows. We consider two or more data flows (arrows in a DFD) between two processes, a process and an external entity or a process and a data store. When bundling data flows, the highest security objective of assets dictates the level of criticality. For non-critical assets, the corresponding data flows must be associated to the same communication channel (e.g., CAN 1) for the bundling to be possible. For instance, in Fig. 1b the flows between the Repair Tool and Edge ECU include assets that are not critical and they are broadcast over the same network. Therefore, they can be bundled. After bundling, the resulting data flow is annotated with a new name and the union of security objectives from both bundled data flows. If the flows contain the same security objective, the highest priority is included in the union.

If any of the data flows considered for bundling contains a high security objective, the area is considered critical and additional criteria need to be met. We must look at the end-to-end flows (as in Fig. 2) for the involved assets. These end-to-end flows must have either the same source, target or both. Otherwise, if they have different source and target, the unaligned parts of the end-to-end flows should not be critical. For instance, in Fig. 2 the data flow VS between Vehicle ECU and Driver Control ECU is marked with a high-priority integrity objective, while WL has a low-priority integrity objective. Therefore, this area in the architecture is considered critical. Both VS and WL data flows are broadcast on the same communication channel. The diagram shows that the VS end-to-end

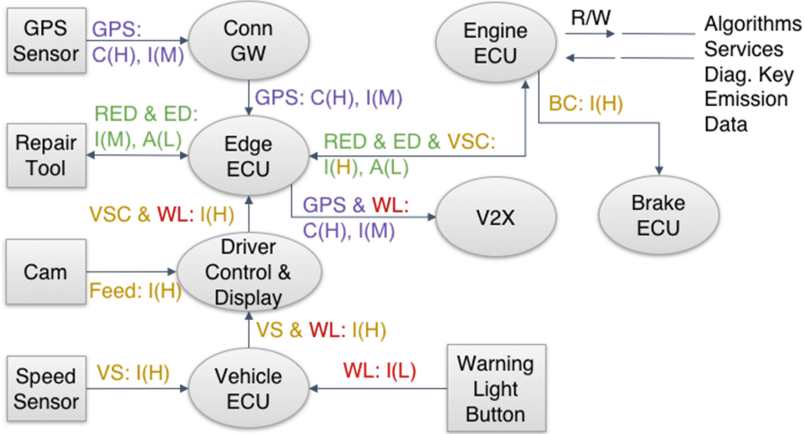


Fig. 4. The abstracted DFD after applying the guidelines.

flows goes from the Speed Sensor to the Driver Control. The WL signal end-to-end flows goes from the Warning Light Button (pressed by the driver) to the Driver Display. Therefore, the assets VS and WL do not have the same source or target. Rather, they only align between the Vehicle ECU and Driver Control. However, the criticality of the non aligned parts is low. In conclusion, the data flows can be bundled according to the more restrictive criteria.

Process folding. Candidate elements for process folding are two adjacent processes. All flows between the candidate processes are considered when determining the criticality of the region.

If the flows do not transport high-priority assets, the region is considered as non-critical. Non-critical processes may be folded if (a) they are not near to a trust boundary and (b) there is a mitigation in place (security assumption) that ensures that at least one of the objectives for the surrounding flows is covered. If a trust boundary is next to the considered region, the processes are likely part of the attack surface and it is considered too risky to bundle them, as relevant threats might be overlooked. In the example of Fig. 2, this guideline applies to the region containing the Driver Control and Driver Display: there are no high priority assets broadcast on ETH 4 and the processes are not near to a trust boundary. Hence, the two processes can be folded.

If any flow between the candidate processes contains a high security objective, the criteria are more restrictive. In addition to the above-mentioned conditions, the processes (c) need to be “mounted” on the same communication channels and (d) there must be mitigations in place ensuring that *all* security objectives over the flows are covered. Unfortunately, no such fold was possible in the running example.

Figure 4 presents the results of the abstraction obtained by applying the guidelines described in this section. Overall, the difference between the original

DFD (in Fig. 1b) and the abstracted DFD (in Fig. 4) lies in the number of flows and processes. The abstracted DFD has 1 process less (Driver Display) and 7 flows less compared to the initial DFD. The simplifications to the model result in a reduced number of identified unimportant threats, as further discussed in Sect. 4.3.

4.2 Effort Reduction During Threat Analysis

Abstracting the architectural model before threat analysis can only take the analyst so far. It is not only the number of DFD elements that makes threat analysis time consuming, but also the type and amount of threat patterns that must be considered for each element. However, the additional information about the assets and the forward assumptions may also be used *during* the analysis to guide the analyst towards the important threats, while omitting the not important ones.

For instance, in Fig. 4, the RED and ED flows have been bundled as a result of the abstraction step. The assets on this bundled flow are ear-marked with the integrity objective (medium priority) and the availability objective (low priority). Therefore, the analyst can focus on the tampering and denial of service threats, and ignore the information disclosure threats¹. As the priorities are not high, the analyst also knows that it is not necessary to dig too deep in the analysis process of this flow. In this respect, the analyst can bring these risk considerations into the process of threat identification and leverage them to reduce the effort spent analyzing this “cool” spot of the system. As a result of the analysis, we found two important threats on that particular flow: physically damaging the OBD port and tampering with the ED signal before it is displayed on the Repair Tool.

Another means of reducing the effort spent eliciting the threats is to use the security assumptions during the threat analysis. For instance, let us suppose we are analyzing the vehicle speed (VS) asset-centric flow described in Fig. 2. The asset is ear-marked with a high-priority integrity objective and, hence, the analyst should carefully consider the potential tampering threats. However, Because of the domain assumption DA1 (“The speed sensor is working securely”) and the domain property DP1 (“Transport-layer security is used between the ECUs”), the tampering threats can be discarded altogether as they are non interesting.

4.3 Effect of Abstraction

The resulting abstracted DFD has been analyzed (by the first author) using the off-the-shelf threat catalogs of STRIDE. This led to the identification of 15 threats, which are not presented here due to space constraints. To appreciate the efficiency of our approach, we remark that a previous study [18] has shown that according to the traditional STRIDE approach, the number of identified threats from a DFD this size should have been around 100. The 15 identified

¹ According to STRIDE, a data flow is subject to three types of threats: tampering (T), information disclosure (I), and denial of service (D).

threats have been reviewed by a security expert (the third author) who routinely performs the threat analysis of automotive systems. The validity and relevance of the identified threats have been confirmed by the expert. Further the expert confirmed that no relevant threats had gone unnoticed.

Our approach resulted in finding less threats because of two reasons. Processes and flows are the elements that are the main cause of threat explosion in a DFD. First, together they make up a large number of architectural elements. Second, they are prone to many types of attacks and, hence, have to be analyzed for several threat categories. Out of a total of six threat categories, STRIDE mandates the consideration of three categories for flows and of all six categories for processes. It is apparent that reducing the number of processes and flows in the DFD can help govern the threat explosion problem. Second, a further reduction of effort is due to a more focused analysis, as explained in Sect. 4.2.

5 Related Work

Significant work has been done in the area of threat analysis and risk assessment (TARA) methods in the automotive domain. Macher et al. [11] recently performed a review of TARA methods in the automotive context. In their main findings, the authors identify most applicable TARA methods for early phase analysis, which closely relate to our approach. The EVITA [2] method is an adaptation of the ISO 26262 HAZOP analysis for security engineering. The method considers potential threats for particular features from the functional perspective by developing attack trees and eventually discovering worse case scenarios. Even though the method employs leveled qualitative risk assessment, no effort is mentioned regarding attack tree minimization. HEAVENS security model [3] analyzes threats based on the STRIDE threat modeling approach and ranks them by assessing the risk with determining the threat, the impact and finally the security level. In contrast to our approach, HEAVENS model is oriented towards ranking the threats only after identifying them. SAHARA [12] method combines the automotive HARA safety analysis with the STRIDE approach to discover impacts of security threats in the safety analysis. The focus of SAHARA lies in understanding the relationship between security threats and safety implications, whereas our work focuses on security aspects only. A security analysis of several applications within a Connected Vehicle Reference Architecture (CVRIA) has been performed by ITERIS and is available online [1]. The published documents include a CIA asset analysis of the V2X communications, while our work focuses on the in-vehicle communications.

Beyond the domain of automotive software, other asset- or software-centric threat modeling approaches are relevant to our work. STRIDE [20] is a popular threat modeling approach, which is based on DFDs. The methodology is comprised of 7 steps: define users and realistic use scenarios, gather assumptions, construct a DFD diagram of the system, map STRIDE to DFD element types, refine threats, document the threats, assign priority via risk analysis

(to counter threat explosion problem) and select mitigations associated to threats. In STRIDE, threat prioritization according to risk value is done after the threat elicitation. Additionally, although STRIDE suggests to start by gathering the security assumptions, no explicit guidance is provided on how to represent and use them in the threat analysis process. Similar to STRIDE, TRIKE [16] is a software-centric methodology. TRIKE includes the identification of assets and actors and offers tool support for attack tree and graph generation. CORAS [10] is an asset-centric methodology for risk analysis consists of a language, a tool and a method. The methodology employs CORAS threat diagrams that describe the threats, vulnerabilities, scenarios and incidents of relevance for the risk in question. Similarly to the aforementioned CORAS methodology, PASTA [23] is an asset-centric, risk-based threat modeling methodology. In PASTA threats are analyzed with the help of use cases and Data Flow Diagrams (DFDs). Further steps are taken for detailed analysis, namely the use of attack trees and abuse cases.

Looking towards recent initiatives to automate threat modeling, our approach relates to the work on extracting threats from DFDs by Berger et al. [6]. Similarly to our work, the authors introduce additional semantics, including the topological behavior. Furthermore, they also develop a set of guidelines, which are used to build a threat model of the architecture. However, these rules are used to discover only cataloged threats and do not aim to handle threat explosion. Perhaps more importantly, our approach differs by analyzing end-to-end assets which, in turn, drives the model abstraction and threat reduction.

Interesting work has been done by Rauter et al. [14] in developing a metric that quantifies software components by their ability to access assets. By doing so, the authors are also able to identify critical areas in the software architecture and consider those for a detailed threat analysis. However, they do not discuss the specifics of how threat analysis techniques benefit from their architectural risk assessment method. Our work also relates to the automated software architecture risk analysis studied by Almosry et al. [5]. The introduced approach is accompanied by a tool and explores security risk analysis by means of formalized signatures of security scenarios and metrics. Similarly, the authors develop a risk-centric architectural analysis approach. However, their work focuses on operationalizing attacks and security metrics to assess the risk level, instead of aiming towards systematically identifying important threats. In addition, the formalized signatures do not seem to consider end-to-end flows of assets.

We also identify several related approaches that could be grouped as attack-centric threat analysis techniques. In these approaches, an explicit model of the attacker is introduced and the analysis is performed from the perspective of an attacker. Examples of such techniques are anti-goals [8], misuse cases [21], misuse case maps [22], abuse cases [13], abuse frames [9], and attack trees [15,19].

6 Discussion and Limitations

The process of creating the enriched model described in Sect. 3 results in a deeper understanding of the system before threat analysis activities take place.

Not only does this contribute towards a common security awareness, but it also enables the identification of realistic threats in the system. As previously mentioned, one of the drawbacks of STRIDE is that the analysis of DFD elements is performed in isolation. In contrast, attack strategies target an asset and often affect a combination of elements. Our approach implicitly considers all the model elements that are related to an asset, which contributes towards detecting attack strategies earlier on in the software development life-cycle. Most importantly, our approach is a step further towards optimizing the effort spent on analyzing threats. This aspect is of primary importance in the industry (especially for complex systems, like CPS) and is often overlooked in the related work. Even though the guidelines are only initial observations, they are synthesized in a domain-independent way, where the main role is played by the semantics of end-to-end flows and not the domain-specific content. Therefore, there is potential for generalizing the guidelines to domains outside the scope of cyber-physical systems. In particular, domains where the software architecture is comprised of different networks and communication protocols may benefit from our approach (e.g., Microservice architecture). However, more investigations have to be made in order to confirm this claim.

Our approach relies on the correctness of the domain assumptions and the truthful representation of the domain properties. This means that the presence of a domain expert is mandatory. Another draw back is that the approach assumes that there are in fact non-critical areas in the architecture. If all the candidate model elements for abstraction have high-priority security objectives, the abstraction may result fewer simplifications, if any at all. Another problem might arise once the amount of end-to-end flows increases. The guidelines do not consider what happens when new scenarios are added. New scenarios might bring along different assets that travel through the same communication channels. As a result, successful abstractions have to be reconsidered. Some of these problems could potentially be alleviated by a tool. In terms of increasing the productivity, the application of guidelines for abstraction can be (semi)automated. Such support may take advantage of our approach, while enabling experts to freely move between abstractions during the analysis. Working towards the automation of threat analysis also caters to security related activities later in the product life-cycle. Notably, after implementation, the planned architecture comes seldom into question again. However, the implemented architecture often differs from the planned architecture too much. Checking for compliance is therefore a complex and important task. As part of future work, we plan to explore ways to synchronize the extended DFDs and the implemented architecture. Furthermore, we acknowledge that we have validated the approach only by illustrating its potential on one simplified architecture. In future work, we will systematically evaluate the benefits of our approach in a series of comparative studies involving both students and industrial experts.

7 Conclusion

In this work, we presented a novel approach for a risk-first security analysis of design artifacts. Without departing too much from well-known techniques like STRIDE, our approach is focused on an end-to-end asset analysis and accommodates forward reasoning on the constraints imposed by the solution space. Our main contributions are (i) a notation to represent end-to-end asset flows and to enrich the analysis models with important security assumptions, and (ii) a set of guidelines for traversing the model and making suitable abstractions. The contributions aim at mitigating the problem of threat explosion, commonly present in STRIDE and other techniques. Additionally, the extended notation supports a more appropriate representation of communication channels, which is valued in the domain of Cyber-Physical Systems. We illustrated our approach by applying it on a simplified system provided by industrial partners in the automotive domain. Preliminary results show that the analysis method indeed yields a reduced number of low priority threats. In future work, we plan to extend the validation of our approach and explore the opportunities for threat analysis automation.

Acknowledgments. This research was partially supported by the Swedish VINNOVA FFI project “HoliSec: Holistic Approach to Improve Data Security”.

References

1. Connected vehicle reference implementation architecture. <http://local.iteris.com/cvria/>. Accessed 25 Aug 2017
2. E-safety vehicle intrusion protected applications. <http://www.evita-project.org/index.html>. Accessed 25 Nov 2016
3. Heavens: Healing vulnerabilities to enhance software security and safety. <http://www.vinnova.se/sv/Resultat/Projekt/Effekta/HEAVENS-HEALING-Vulnerabilities-to-ENhance-Software-Security-and-Safety/>. Accessed 25 Nov 2016
4. HoliSec: Holistiskt angreppssätt att förbättra datasäkerhet. <http://www2.vinnova.se/sv/Resultat/Projekt/Effekta/2009-02186/HoliSec-Holistiskt-angreppssatt-att-forbatta-datasakerhet/>. Accessed 14 June 2017
5. Almorsy, M., Grundy, J., Ibrahim, A.S.: Automated software architecture security risk analysis using formalized signatures. In: Proceedings of the 2013 International Conference on Software Engineering, pp. 662–671. IEEE Press (2013)
6. Berger, B.J., Sohr, K., Koschke, R.: Automatically extracting threats from extended data flow diagrams. In: Caballero, J., Bodden, E., Athanasopoulos, E. (eds.) ESSoS 2016. LNCS, vol. 9639, pp. 56–71. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30806-7_4
7. Howard, M., Lipner, S.: The Security Development Lifecycle, vol. 8. Microsoft Press, Redmond (2006)
8. van Lamsweerde, A.: Elaborating security requirements by construction of intentional anti-models. In: Proceedings of the 26th International Conference on Software Engineering, ICSE 2004, pp. 148–157. IEEE Computer Society, Washington, DC (2004). <http://dl.acm.org/citation.cfm?id=998675.999421>

9. Lin, L., Nuseibeh, B., Ince, D., Jackson, M.: Using abuse frames to bound the scope of security problems. In: Proceedings 12th IEEE International Requirements Engineering Conference, pp. 354–355. IEEE (2004)
10. Lund, M.S., Solhaug, B., Stølen, K.: Model-Driven Risk Analysis: The CORAS Approach. Springer Science & Business Media, Heidelberg (2010)
11. Macher, G., Armengaud, E., Brenner, E., Kreiner, C.: A Review of threat analysis and risk assessment methods in the automotive context. In: Skavhaug, A., Guiochet, J., Bitsch, F. (eds.) SAFECOMP 2016. LNCS, vol. 9922, pp. 130–141. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45477-1_11
12. Macher, G., Sporer, H., Berlach, R., Armengaud, E., Kreiner, C.: Sahara: a security-aware hazard and risk analysis method. In: 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 621–624. IEEE (2015)
13. McDermott, J., Fox, C.: Using abuse case models for security requirements analysis. In: Proceedings 15th Annual Computer Security Applications Conference, (ACSAC 1999), pp. 55–64. IEEE (1999)
14. Rauter, T., Kajtazovic, N., Kreiner, C.: Asset-centric security risk assessment of software components. In: 2nd International Workshop on MILS: Architecture and Assurance for Secure Systems (2016)
15. Saini, V., Duan, Q., Paruchuri, V.: Threat modeling using attack trees. *J. Comput. Sci. Coll.* **23**(4), 124–131 (2008)
16. Saitta, P., Larcom, B., Eddington, M.: Trike v. 1 methodology document [draft] (2005). http://dymaxion.org/trike/Trike_v1_Methodology_Documentdraft.pdf
17. Scandariato, R., Walden, J., Joosen, W.: Static analysis versus penetration testing: a controlled experiment. In: 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), pp. 451–460. IEEE (2013)
18. Scandariato, R., Wuyts, K., Joosen, W.: A descriptive study of Microsoft’s threat modeling technique. *Requir. Eng.* **20**, 163–180 (2015)
19. Schneier, B.: Attack trees. *Dr. Dobb’s J.* **24**(12) (1999)
20. Shostack, A.: Threat Modeling: Designing for Security. Wiley, Indianapolis (2014)
21. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. *Requir. Eng.* **10**(1), 34–44 (2005). <https://doi.org/10.1007/s00766-004-0194-4>
22. Tøndel, I.A., Jensen, J., Røstad, L.: Combining misuse cases with attack trees and security activity models. In: International Conference on Availability, Reliability, and Security, ARES 2010, pp. 438–445. IEEE (2010)
23. UcedaVelez, T., Morana, M.M.: Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis. Wiley, Hoboken (2015)
24. Van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software, vol. 10. Wiley, Chichester (2009)
25. Wuyts, K., Scandariato, R., Joosen, W.: Empirical evaluation of a privacy-focused threat modeling methodology. *J. Syst. Softw.* **96**, 122–138 (2014)
26. Yu, H., Lin, C.W.: Security concerns for automotive communication and software architecture. In: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPs), pp. 600–603. IEEE (2016)

A Methodology to Assess Vulnerabilities and Countermeasures Impact on the Missions of a Naval System

Bastien Sultan¹(✉), Fabien Dagnat², and Caroline Fontaine³

¹ Chair of Naval Cyber Defense, École Navale – CC 600,
29240 Brest Cedex 09, France

`bastien.sultan@ecole-navale.fr`

² IRISA, IMT Atlantique, UMR 6074, CS 83818, 28238 Brest Cedex 03, France

`fabien.dagnat@imt-atlantique.fr`

³ CNRS, IMT Atlantique, UMR 6285, Lab-STICC, CS 83818,
28238 Brest Cedex 03, France

`caroline.fontaine@imt-atlantique.fr`

Abstract. The paper proposes a methodology to assess the impacts of vulnerabilities, attacks and countermeasures on a vessel's missions, and a metric designed to express and compare these impacts. A behavioral modeling approach for depicting naval systems and missions is presented. Then the paper introduces a model-checking based impact assessment method. The cyber events are integrated in the behavioral model through model mutation. Then, for each mission, their impact is computed by performing a series of model checks. The paper also discusses the algorithmic complexity of the impact assessment method.

1 Introduction

Ships are complex cyber-physical systems operated to perform a variety of missions. Due to the increasing number of cyber assets among their components, these systems can be affected by vulnerabilities which, when exploited, can impact their behavior. Therefore it is crucial to apply countermeasures in order to mitigate these vulnerabilities, and fundamental to ensure that deployed patches does not negatively affect the system's dependability and functionality. This problem is known as patch management and has been discussed in several books and papers, including [5, 7].

The work we present in this paper is part of a project aiming to design a patch management process applied to naval systems. This process has been described in [13], and its purpose is to provide a comparative evaluation of vulnerabilities and available countermeasures, based on the impact they can have on the system's ability to fulfill its missions. The first phase of this process extends over the entire life cycle of the vessel and is intended to build behavioral models of the system and its missions, and to keep them up-to-date. This task requires to gather

and federate different types of models, produced by different entities: naval doctrine, missions, functional and physical architecture, etc. Timed automata and safety properties are then deduced from this federation. The automata model the vessel's behavior and its missions, and the properties are used to validate the correct behavior of the automata through model-checking. Concurrently with this task, a vulnerability monitoring is performed over the life cycle of the system.

The second task of the patch management process is triggered by a vulnerability discovery. First, existing and/or specifically designed countermeasures are gathered. Automata mutation are then performed in order to produce a set of behavioral models of the vulnerable system and the patched system. Several model checks are then led in order to measure the ability of the vulnerable and patched systems to fulfill the missions set. These behavioral models are then composed with attack automata and a second set of model checks is led. From these two series of verifications, impact metrics are deduced and available countermeasures are ordered according to their efficiency and their innocuousness.

The main scientific issue raised by such a process is the impact assessment problematic: how to quantify an impact? What metric can be used in order to compare them? These questions will be answered in the following sections of this paper. Before a review of the related works regarding impact assessment we will introduce the context and some key definitions. The third part of the paper will then focus on the core of our work: an impact assessment methodology based on formal models verification. This section will present our modeling approach: the different classes of models we need to gather, and the system modeling through finite state automata. The concept of automata mutation will be defined, prior to describe the impact assessment methodology and impact comparison metric. Then in the last part we will discuss the algorithmic performance of this method.

2 Context

Ships are complex systems composed of several subsystems – several of which are cyber-physical systems (CPS) – operated by a crew in order to fulfill a set of missions.

The functions of a naval system involve a lot of different entities. For instance the navigation process relies on GNSS sensors and/or inertial units, data processing systems, but also steering and propulsion systems. The security of these equipments is essential: a malfunction of one of them can lead to endanger the vessel and reduce its ability to accomplish its missions. Yet weaknesses can affect various elements (sensors, actuators, ...) and disturb several steps of these missions. Therefore it is essential to mitigate these weaknesses. Nevertheless, the countermeasures intended to do so can also disturb the ship's behavior. An assessment of the impact of such vulnerabilities and countermeasures is thus crucial. We need firstly to define the following concepts: *system*, *vulnerability*, *countermeasure* and *impact* (Fig. 1).

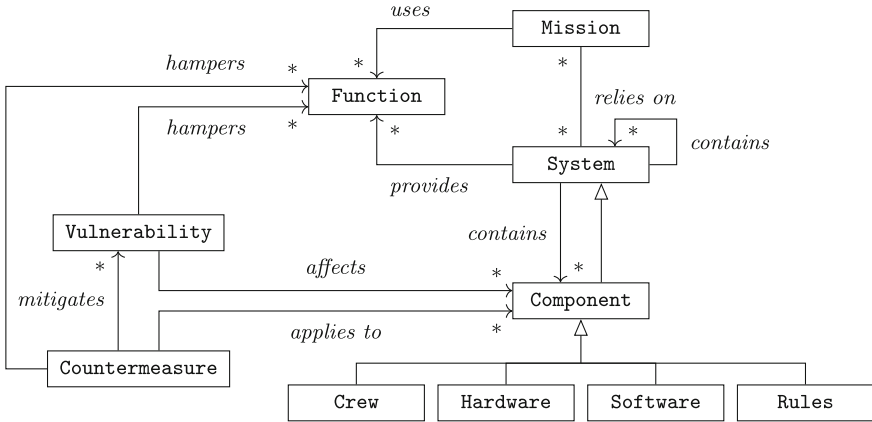


Fig. 1. Metamodel of a (naval) system

- **System:** “A composite, at any level of complexity, of personnel, procedures, materials, tools, equipment, facilities, and software. The elements of this composite entity are used together in the intended operational or support environment to perform a given task or achieve a specific production, support, or mission requirement” [12]. A system has a security policy [1] whose aim is especially to ensure the system’s safety.
- **Vulnerability:** “A weakness in the physical layout, organization, procedures, personnel, management, administration, hardware, firmware or software of a system that may be exploited” to violate its security policy [10].
- **Countermeasure:** A countermeasure is “any action, device, procedure, technique, or other measure” that mitigates a given vulnerability [10].
- **Impact:** The impact of a vulnerability (resp. a countermeasure, an attack) represents the consequences of a vulnerability (resp. a countermeasure, an attack) on the system’s ability to safely fulfill its missions. This metric allows to compare the consequences of two countermeasures, a countermeasure and a vulnerability, etc. It is therefore a key metric to choose the more efficient and the less hazardous countermeasure to mitigate a vulnerability.

3 Related Works

The patch management problem has been addressed in several papers, including [4, 5, 7]. B. Brykczynski and R. A. Small introduce in [4] eight patch management key practices, among others *Maintain awareness of IT infrastructure* (configuration management), *Monitor vulnerability alerts*, *Assess and respond to vulnerability alerts* (assess the impact of a potential attack, evaluate the attack surface, conceive specific countermeasures if no patch is available, ...), *Test and evaluate patches* (are the patches effective? Will they result in intolerable side effects?) and *Install patches*.

We find similar activities in the process established in [5], which method relies on five consecutive steps: 1. *Vulnerability Information Receiving*, 2. *Impact Assessment*, 3. *Test and Deployment Planning*, 4. *Testing* and 5. *Automatic Patch Deployment and Auditing*. We can notice that the *Impact Assessment* step is mainly a scoping task, aiming to identify the assets affected by the vulnerability. The impact of patches is assessed in tasks 3 and 4, through a testing approach.

Recommended Practices for Patch Management of Control Systems [7] notably highlights the importance of a configuration management program and patch testing. Testing should be led in an environment “*that closely simulates the operational environment*” and should allow to verify if the patch effectively mitigates the vulnerability, and if the patched application remains functional. These three papers insist on the interest of patch testing, in order to ensure the effectiveness and the innocuousness of the tested patches. However they provide neither a practical methodology to assess the impacts of patches and vulnerabilities, nor a metric to evaluate these impacts.

Impact assessment of cyber events is treated in various papers. Gonzalez-Granadillo et al. [8] “*provide a method to calculate the impact of cyber attacks and security countermeasures*” using a geometric model. “*Services, attacks and countermeasures [are represented] in an n-dimensional coordinate system, n being the number of dimensions (e.g., user account, channel, resource, etc.)*”. As a result, this method allows to determine the impacts of attacks on the system, as well as the mitigation level, residual risks and potential collateral damages of patches. However, these impacts are not expressed in relation to the system’s missions.

Scott Musman et al. introduce in [11] a methodology for evaluating the impact of cyber attacks on missions. Their approach consists in the following steps: model the mission through BPMN standard (for each mission, a new model of the assets and their interactions is built); execute the model and estimate the measures of effectiveness (MOE1); transform the mission model to take into consideration the attack effects on the mission; execute the transformed model and compute the measures of effectiveness (MOE2): the impact is then given by the difference between MOE1 and MOE2. This methodology should probably be suitable for patch impact assessment. However it doesn’t models the components used outside of the mission context, which can be used to propagate an attack to mission critical assets.

Gabriel Jakobson [9] “*proposes a conceptual framework and a method for assessing impact that cyber attacks might have to cyber assets, services, and missions*”. The system (cyber terrain) is modeled as a directed graph composed of three kinds of vertices: hardware, software and services. The edges of this graph represent dependency between components (connectivity, containment, location, etc.). The cyber terrain is a dynamic structure: “*its components and their interdependencies are a function of time*”. Moreover, each vertex of the graph has a dynamic operational capacity $OC \in [0, 1]$. Missions are modeled as sequences of steps (which can be tasks or flows of tasks) and an impact dependency graph defines the dependencies between each step of the mission and the cyber terrain.

Attack models are parametrized by their impact factor $IF \in [0, 1]$ “indicating to what degree the attack is capable to compromise the attacked asset”. The OC of an attacked asset decreases depending on the IF of the attack. The impact of an attack on a given mission is assessed in three steps: 1. the OC of the attacked assets are computed; 2. the OC of the assets that could be affected by a propagation of the attack are computed; and 3. the OC of the mission is then computed depending on the OC previously computed.

Our approach also uses distinct models to depict the system, its missions and their interactions, but the modeling approach is different since we focus on the behavior of the components. An attack, a countermeasure deployment or a vulnerability disclosure is modeled by an alteration of the behavioral model. Behavioral modeling is common in the field of complex systems modeling [2, 3], and the alteration of such models has been explored in several papers, including [2]. In this paper the authors propose a model-based mutation testing approach based on timed automata. The aim of the method is to determine a set of model mutants reflecting an incorrect implementation of a device under test. Given a correct model of the device, “a faulty implementation and a test-case failing on the implementation”, they generate all the possible mutants from the initial model, and then execute several filtering steps in order to eliminate the mutants that not “show any faulty behavior along the path of the test case”. Finally, the remaining subset of mutants corresponds to the possible wrong implementation models. The paper also defines mutation operators (change target, change source, change guard, negate guard, etc.) applied to create “first-order mutants”, *i.e.* “one operator at a time [is applied] to one part of the model at a time”. Our mutation process presents some differences with this one, insofar as several of our mutation operators are different and we apply several of them at a time; moreover we generate mutants from a description (of an attack, a vulnerability, a countermeasure) instead of generating every possible mutants.

4 A Methodology for Impact Assessment Based on Timed Automata

4.1 Overall Presentation

Modeling the System. As said in the introduction, our impact assessment method relies on a behavioral modeling of the system. We can model the system’s behavior and missions through timed automata. Indeed, missions and behaviors of the system’s components can be seen as state sequences (*cf.* Fig. 2). The dependency link between system’s assets and missions can be depicted through this kind of modeling. For instance, the guard between `shipMoving` and `measurementCampaign` in Fig. 2(a) ensures that the fourth step of the mission should only be reached if the rudder can rotate. Therefore, we can model our system and its missions through two classes of automata:

1. *System Automata*, modeling the ship’s behavior. System Automata are timed, in order to model the operating times of the system’s assets;

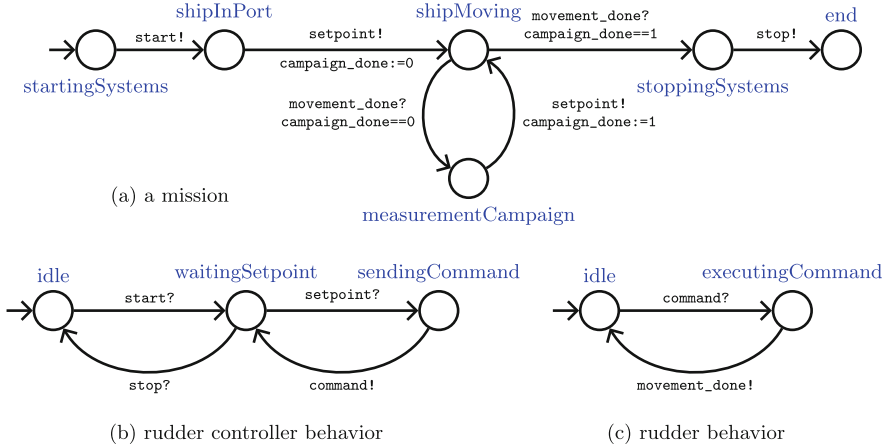


Fig. 2. Some example automata

2. *Mission Automata*, describing the mission steps.

We then need to express a set of properties allowing to check the correct behavior of the system through a model-checking approach. We can distinguish three groups of properties:

- A: the system can fulfill its missions;
- T: the system can fulfill its missions within the correct period;
- I: the system can fulfill its missions while preserving the integrity of its components.

In the following subsection, we will see how to deduce these automata and the related properties.

Assessing the Impact of a Cyber Event. For each cyber event (vulnerability, cyber-attack or countermeasure), the changes on the “real” system are modeled by mutating the *System Automata*. We use mutations

Definition 1 (mutation). *Let \mathcal{A} be an automaton such as defined in [2]. We define a mutation of \mathcal{A} as any add or removal of states, transitions, clocks, guards, or any change in the value of its guards or clocks, resulting in an automaton \mathcal{A}' .*

Example 1. We can imagine that the rudder controller is vulnerable to a DOS attack. The related mutation can be {add a state blocked; add a transition from idle to blocked triggered by a synchronisation attack?; add a transition from waitingSetpoint to blocked triggered by a synchronisation attack?; add a transition from blocked to blocked triggered by a synchronisation setpoint?}. As a result we obtain the automata depicted in Fig. 3.

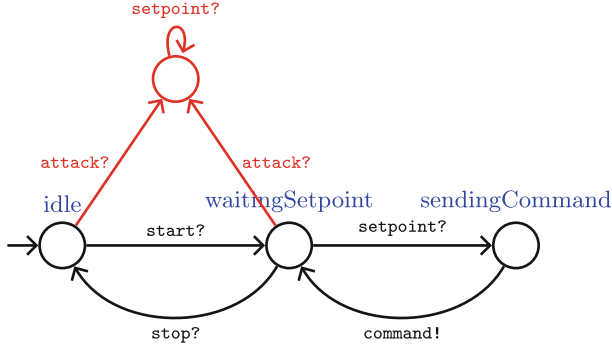


Fig. 3. Vulnerable rudder controller's behavior

Applying the mutations on the *System Automata*, we can model the comportamental effect of any vulnerability, attack or countermeasure at the subsystem level. Then, for each mission, we compose the mutated *System Automata* with the mission automata and we check the model through the different properties: from this basis we can assess the overall impact on the system.

Subsections 4.3 and 4.4 will detail the impact assessment sequence within the context of the Patch Management process, and will introduce a metric intended to express and compare the impacts.

4.2 Models Federation

In order to build system and mission automata, such as the set of properties, we federate two classes of models: context models, generic to all vessels, and project-specific models, specific to a given ship (or a class of ships). From this basis we obtain six kinds of federated models allowing to build system and missions automata (*cf.* Fig. 4).

These models and the automata are kept up-to-date throughout the whole system's life-cycle to maintain an up-to-date configuration management database.

4.3 Impact Assessment Sequence

Figure 5 illustrates the impact assessment methodology.

1. Before the discovery of a vulnerability, we have an inaccurate view of the system, modeled by the *nominal system automata* \mathcal{A}^n . \mathcal{A}^n verifies \mathcal{P} , the set of properties.

Example 2. We will focus on three components of a ship: a rudder, the rudder controller, and a network switch allowing the crew to access the Internet; and two missions: the mission depicted on Fig. 2(a) (mission 1), and a mission consisting in reporting the measurement's campaign results through the Internet

	Context Models	Project-specifics Models	Federated Models	Automata
Components	Components description	Components description	Components description	Components Automata
System	N/A	Components list / Network topology / Functions list / Workflow diagrams	System architecture	Nominal System Automata (\mathcal{A}^n)
Missions	Naval doctrine	Missions list	Missions list, sorted by priority	Missions Automata – Properties (\mathcal{P})
Vulnerability	Information gathered from CERT	Filtered information	Mutations list	Vulnerable System Automata (\mathcal{A}^v)
Countermeasures	Information gathered from CERT	Filtered information / specifically designed countermeasures	Mutations list	Patched System Automata (\mathcal{A}_i^p)
Attacks	Published exploits	Specifically designed attacks	Attack trees	Attack Automata – Attacked System Automata ($\mathcal{A}_k^{v/att}$, $\mathcal{A}_{i,k}^{p/att}$)

Fig. 4. Models federation

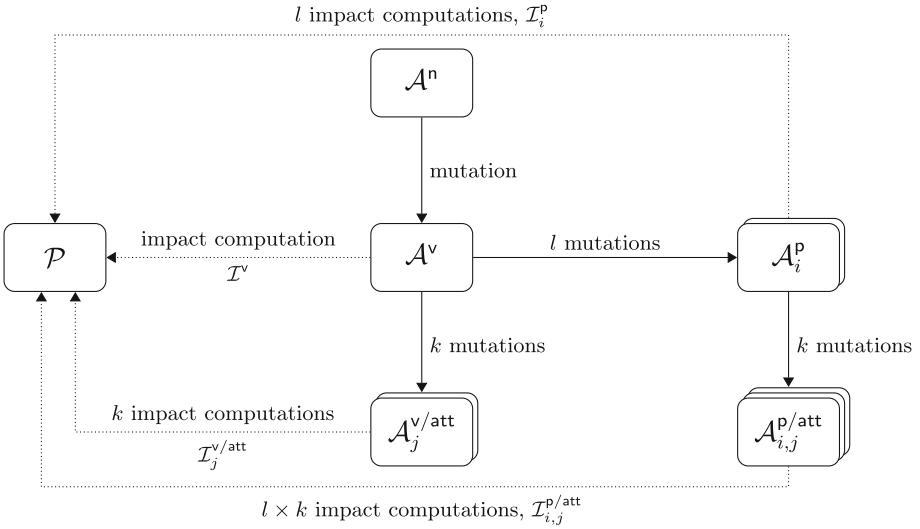


Fig. 5. Impact assessment process for patch management

(mission 2). The network switch is modeled by an automaton setting a variable *internet_link* to 1 when it connects to the Internet. \mathcal{A}^n is the composition of the automata depicted in Fig. 2(b) and (c), and the network switch automaton. \mathcal{A}^n verifies the following set \mathcal{P} : {The state *end* of mission 1 automata can be reached; The state *end* of mission 1 automata can be reached within x seconds; The state *end* of mission 2 automata can be reached; The state *end* of mission 2 automata can be reached within x seconds; The rudder is operational; The controller is operational; The network switch is operational}.

2. At the discovery of a vulnerability modifying the ship's behavior, \mathcal{A}^n becomes a wrong modeling and we need to change it. We apply a list of mutations leading to \mathcal{A}^v , a *vulnerable system automata* modeling the vulnerable system. We note \mathcal{I}^v the impact of the vulnerability.

Example 3. We can now imagine that the controller is affected by a remote DOS vulnerability such as the one introduced in 3.1. We apply the related mutation to obtain \mathcal{A}^v , the composition of the automata depicted in Figs. 2(c) and 3, and the network switch automaton.

3. For each countermeasure c_i , for i in $[1..l]$, we would like to deploy, we model its effect on the vulnerable system by the mutation of \mathcal{A}^v into a *patched system automata* \mathcal{A}_i^p .

We note \mathcal{I}_i^p the impact of the countermeasure c_i . The difference between \mathcal{I}^v and \mathcal{I}_i^p allows to measure the non-regression test of the countermeasure c_i .

Example 4. We can imagine that two countermeasures are available: c_1 , which consists in a software patch without any behavioral modification, and c_2 , which consists in forbidding the network switch to establish the internet connexion. \mathcal{I}_1^p and \mathcal{I}_2^p represent their respective impacts.

4. Then, for each attack att_j , for j in $[1..k]$, exploiting the vulnerability we want to patch, we compose \mathcal{A}^v and $\{\mathcal{A}_i^p\}_{i \in [1..l]}$ with an attack automaton modeling the attack sequence. We obtain the *vulnerable system under attack automata* $\mathcal{A}_k^{v/\text{att}}$ and the set of *patched system under attack automata* $\{\mathcal{A}_{i,j}^{p/\text{att}}\}_{i,j \in [1..l] \times [1..k]}$.

We note $\mathcal{I}_j^{v/\text{att}}$ the impact of the attack att_j on the vulnerable system, and $\mathcal{I}_{i,j}^{p/\text{att}}$ its impact on the patched system where the countermeasure c_i is deployed. The difference between $\{\mathcal{I}_j^{v/\text{att}}\}_{j \in [1..k]}$ and $\{\mathcal{I}_{i,j}^{p/\text{att}}\}_{j \in [1..l] \times [1..k]}$ allows to measure the effectiveness test of the countermeasure c_i .

Example 5. Then we compose the vulnerable system automata with the attack automata depicted in Fig. 6, corresponding to a remote attack att_1 . The first transition of this automata can only be fired if the network switch is able to establish the internet connexion, modeled by the integer *internet_link*. The impact matrix of the vulnerable system under attack is $\mathcal{I}_1^{v/\text{att}}$. By composing the patched system automata \mathcal{I}_1^p and \mathcal{I}_2^p with the attack automata, we deduce the impact matrices $\mathcal{I}_{1,1}^{p/\text{att}}$ and $\mathcal{I}_{2,1}^{p/\text{att}}$.

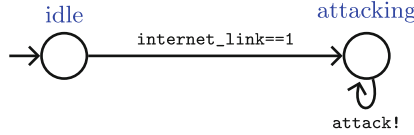


Fig. 6. Attack automaton

At the end of this impact assessment process, we can compare the non-regression and the effectiveness of each countermeasure.

4.4 Modeling and Comparing Impacts

We have decided to evaluate the impact through model checking. The computation works as follows: for each mission, we compose the system automata with the mission automaton; then, we check the properties related to this mission (the mission can be fulfilled and the mission can be fulfilled within the correct period) and the properties related to the integrity of every component. As a result, we check $m + 2$ properties per mission, where m stands for the number of components in the system. The model checking results are stored in an *impact matrix*. Each line of this matrix corresponds to one property, while each column corresponds to one mission. The columns are sorted according to the criticality of each mission, from the more critical to the less.

$$\begin{pmatrix} A_0 & A_1 & \cdots & A_{n-2} & A_{n-1} \\ T_0 & T_1 & \cdots & T_{n-2} & T_{n-1} \\ I_{0,0} & I_{0,1} & \cdots & I_{0,n-2} & I_{0,n-1} \\ I_{1,0} & I_{1,1} & \cdots & I_{1,n-2} & I_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I_{p,0} & I_{p,1} & \cdots & I_{p,n-2} & I_{p,n-1} \end{pmatrix}$$

The values of this matrix belong to $\{0, 1\}$: 1 if the property is verified, and 0 otherwise. Each A_k corresponds to the attainability of the final state of the mission k , T_k to the attainability of the final state of the mission k within the correct period, $I_{j,k}$ to the ability to fulfill the mission k while preserving the integrity of the component j .

As the importance of a given property depends on the context of each mission, we then define a permutation σ on the elements of each column. For instance, the integrity of a bathymetric sonar will be more important than the integrity of a thermosalinograph in the context of a bathymetric survey campaign, while the thermosalinograph integrity will be more important in the context of a salinity survey campaign. This permutation will sort the elements according to the importance of their related property, from the more important to the less. We obtain the filtered impact matrix:

$$(\sigma(\mathcal{C}_0) \sigma(\mathcal{C}_1) \cdots \sigma(\mathcal{C}_{n-1}))$$

where $\mathcal{C}_0 \dots \mathcal{C}_n$ are the columns of the impact matrix. The main interest of expressing the different impacts through these matrices is to facilitate their comparison. Indeed, since columns and lines are sorted in descending order, we can easily ordinate the different filtered impact matrix by sorting them in lexicographical order.

Definition 2 (lexicographical order). Let $(a_{i,j}), (b_{i,j}) \in \{0, 1\}^{m \times n}$. $(a_{i,j}) <_{lex} (b_{i,j}) \iff \exists (k, l) \in \llbracket 0, m \rrbracket \times \llbracket 0, n \rrbracket \mid (\forall p, q \in \llbracket 0, m \rrbracket \times \llbracket 0, l - 1 \rrbracket, a_{p,q} = b_{p,q}) \wedge (a_{0,l} = b_{0,l} \wedge \dots \wedge a_{k-1,l} = b_{k-1,l}) \wedge (a_{k,l} < b_{k,l})$.

Example 6. Considering that our filtered impact matrices will gather, in this order, the following properties:

Mission 1	Mission 2
The state <i>end</i> of mission 1 automata can be reached	The state <i>end</i> of mission 2 automata can be reached
The state <i>end</i> of mission 1 automata can be reached within x seconds	The state <i>end</i> of mission 2 automata can be reached within x seconds
The rudder is operational	The network switch is operational
The controller is operational	The controller is operational
The network switch is operational	The rudder is operational

Then the impacts $\mathcal{I}^v, \mathcal{I}_1^p, \mathcal{I}_2^p, \mathcal{I}_1^{v/att}, \mathcal{I}_{1,1}^{p/att}$ and $\mathcal{I}_{2,1}^{p/att}$ will be given by:

$$\mathcal{I}^v = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \quad \mathcal{I}_1^p = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \quad \mathcal{I}_2^p = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

$$\mathcal{I}_1^{v/att} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} \quad \mathcal{I}_{1,1}^{p/att} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \quad \mathcal{I}_{2,1}^{p/att} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Comparing these matrices, we notice that $\mathcal{I}_1^p = \mathcal{I}^v$ and $\mathcal{I}_2^p < \mathcal{I}^v$. This indicates that the countermeasure c_1 will probably cause no regression, unlike the countermeasure c_2 . Then, we can see that $\mathcal{I}_{1,1}^{p/att} > \mathcal{I}_1^{v/att}$ and $\mathcal{I}_{2,1}^{p/att} > \mathcal{I}_1^{v/att}$: both countermeasures mitigate the vulnerability.

5 Discussion About Complexity

Combinatorial explosion is a well-known issue in complex systems model-checking [6]. Given the number of subsystems in a modern vessel, we have to take this problem into consideration. The complexity of our impact assessment will depend on two parameters: the number of model checks to lead per vulnerability, and the size of the automata we want to check.

We can estimate at least the first of these parameters: considering our system has n missions and m components, each impact matrix computation will lead to $n \times (m + 2)$ properties verifications. Then, according to Fig. 5 each vulnerability implies $(1 + l + k + k \times l)$ impact matrix computations. Therefore a vulnerability disclosure will result in $n \times (m + 2) \times (1 + l + k + k \times l)$ model checks. If we postulate that our vessel has 20 missions, 2000 components, and that for each vulnerability 5 different attacks and 5 different patches are available, then a vulnerability disclosure will lead to 1 441 440 model checks.

The size of the checked automata will depend on the modeled system and missions. For the purpose of our research we have modeled two subsystems of a fictitious oceanographic survey vessel: propulsion subsystem (a SCADA HMI (human-machine interface), three PLC (programmable logic controllers), a reverse/reduction gearbox, a diesel engine, a gas turbine and a rudder) and bathymetric survey subsystem (a SCADA HMI, a PLC and a multibeam sonar sensor). This system has been modeled in FIACRE [3] and compiled with OBP tool [6]. We have then modeled two mission automata: a first one close to Fig. 2(a), using both subsystems of the vessel, and a second one using only the propulsion subsystem. After composing this model with both missions we can notice that the complexity of the resulting automaton is variable: for the composition with mission 1, OBP computes a 39 069 states and 144 756 transitions resulting automaton, while the composition with mission 2 gives a 4 095 states and 11 380 automaton. At this stage of our research we did not face state-space explosion, but it will probably happen with real systems models. In order to reduce the resulting exploration time, we could consider the following options:

1. Lazy evaluation. Since the interest of impact assessment in the context of a patch management process is to compare the impacts of the different counter-measures, we can compute and compare the matrices in parallel. As soon as one matrix “becomes” less than the others it is eliminated from the process.
2. Partial computation. In addition to lazy evaluation, we can restrict the set of studied missions to the few most critical ones. This will reduce the number of columns of the matrices, therefore the number of model verifications to lead.
3. High performance computing. Checking our models with supercomputers will also improve the exploration time. Since a patch management process applied to complex vessels should be led by large organizations, this seems to be a realistic option.

6 Conclusion

Patch management of mission-critical systems is a delicate process insofar as countermeasures can introduce side-effects compromising the missions. Moreover, the available countermeasures have different mitigation levels. At a vulnerability discovery, both side-effects and mitigation levels have to be taken into account in order to select the more efficient and the less harmful countermeasure.

We have proposed in this paper a methodology to assess vulnerability, countermeasures and attacks impacts on a vessel's missions. This methodology is based on a series of model checks performed on a behavioral model of the system. Prior to the impact computation of a given cyber event, mutations of the model are led in order to depict the behavioral effect of the cyber event on the system. Model checks are then performed and the impact of the cyber event is expressed through the impact metric introduced in Sect. 4.4. This metric allows, in the context of a Patch Management process, to order countermeasures depending on their potential side-effects and their mitigation levels. We have then discussed the algorithmic complexity of our method, and proposed several options to reduce it.

First experiments have been led on the basis of fictitious systems, and further work will focus on applying our methodology to real systems, in order to give a more accurate estimation of the behavioral models complexity. The subsets of properties (A , T , I) will also be completed with other classes of properties aiming to measure the quality of the mission (for instance the percentage of fulfilled tasks, the quality of survey data for a scientific mission, etc.).

Acknowledgment. The work presented in this paper is funded by Brittany regional council, as part of the project SPANS.

We also thank Philippe Dhaussy, from ENSTA Bretagne, for his assistance and advice.

References

1. Abou El Kalam, A.: Security of critical infrastructures and networks. Habilitation à diriger les recherches, Institut National Polytechnique de Toulouse - INPT, December 2009
2. Aichernig, B.K., Hörmaier, K., Lorber, F.: Debugging with timed automata mutations. In: Bondavalli, A., Di Giandomenico, F. (eds.) SAFECOMP 2014. LNCS, vol. 8666, pp. 49–64. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10506-2_4
3. Berthomieu, B., Bodeveix, J.-P., Farail, P., Filali, M., Garavel, H., Gauffillet, P., Lang, F., Vernadat, F.: Fiacre: an intermediate language for model verification in the Topcased environment. In: ERTS 2008, Toulouse, France, January 2008
4. Brykczynski, B., Small, R.A.: Reducing internet-based intrusions: effective security patch management. *IEEE Softw.* **20**(1), 50–57 (2003)
5. Chang, C.-W., Tsai, D.-R., Tsai, J.-M.: A cross-site patch management model and architecture design for large scale heterogeneous environment. In: 2005 39th Annual 2005 International Carnahan Conference on Security Technology, CCST 2005, pp. 41–46, October 2005

6. Dhaussy, P., Roger, J.C., Boniol, F.: Reducing state explosion with context modeling for model-checking. In: 2011 IEEE 13th International Symposium on High-Assurance Systems Engineering, pp. 130–137, November 2011
7. U.S. DHS National Cyber Security Division: National Cyber Security Division, Recommended Practice for Patch Management of Control Systems. December 2008
8. Gonzalez-Granadillo, G., Garcia-Alfaro, J., Debar, H.: A polytope-based approach to measure the impact of events against critical infrastructures. *J. Comput. Syst. Sci.* **83**(1), 3–21 (2017)
9. Jakobson, G.: Mission cyber security situation assessment using impact dependency graphs. In: 14th International Conference on Information Fusion, pp. 1–8, July 2011
10. Jaworski, L.M.: Tandem threat scenarios: a risk assessment approach. In: Proceedings of the 16th National Computer Security Conference, pp. 155–164 (1993)
11. Musman, S., Temin, A., Tanner, M., Fox, D., Pridemore, B.: Evaluating the impact of cyber attacks on missions. In: International Conference on Cyber Warfare and Security, p. 446. Academic Conferences International Limited (2010)
12. U.S. Department of Defense: Department of Defense Handbook: System Security Engineering - Program Management Requirements, MIL-HDBK-1785 (1995)
13. Sultan, B., Dagnat, F., Fontaine, C.: Maîtrise des correctifs de sécurité pour les systèmes navals. In: CIEL 2016: 5ème Conférence en Ingénierie du Logiciel, pp. 1–6 (2016)

STRIDE to a Secure Smart Grid in a Hybrid Cloud

Bojan Jelacic^(✉), Daniela Rosic, Imre Lendak, Marina Stanojevic,
and Sebastijan Stoja

Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia
{bojan.jelacic,drosic,lendak,marina.stanojevic,
sebastijan.stoja}@uns.ac.rs

Abstract. This paper describes one possible migration scenario of Smart Grid Industrial Control System (ICS) elements to the computing cloud while maintaining the existing level of information system security. We performed a software centric threat analysis of the Smart Grid ICS, i.e. the most important elements of the system were analyzed following the STRIDE methodology. Security risks were analyzed based on the combined effects of the likelihood of a successful attack and the impact on the identified critical components of the Smart Grid ICS. Risk matrices were used to determine the measure of the security risk. Based on our threat analysis we propose a migration scenario to a hybrid (community & private) cloud. In our scenario, the ICS elements with higher risk tolerance were deployed in a community cloud, while the elements with lower risk tolerance were kept on premise in a private cloud.

Keywords: Industrial Control System (ICS) · Smart Grid · Cloud computing
Software centric threat analysis · STRIDE · Cyber security · Risk assessment

1 Introduction

It is common knowledge that the number of electricity consumers is continuously increasing and this trend will probably continue in the future. Existing energy networks are not able to supply this increasing demand without significant investments in infrastructure and automated computer systems. The implementation of the latest advances in the information and communication technologies can turn the power networks into truly Smart Grids, which is controlled and partially automated via a Smart Grid Industrial Control System (ICS). Cloud computing is one key enabling technology on a roadmap to the future Smart Grids, as it allows system operators to provision only the necessary computing, communication and storage resources and thereby lower costs. The migration to a computing cloud is a considerable challenge, both because of multiple decade-long reliance on closed and utility-owned computing resources and its possible impact on information security. Hackers might turn off electricity or otherwise deny the critical service offered by companies in the electric power sector (e.g. the cyberattacks against Ukrainian distribution system operators), or they might steel sensitive information about customers. Although the above types of real and theoretical attacks might seem to be strong deterrent against migrating any of the Smart Grid ICS services to the computing cloud, we will argue in this paper that such migration is possible without significantly

affecting the current level of information security, i.e. without compromising the system operator's security posture. We will base that claim on a security assessment of the most relevant ICS components performed by following Microsoft's Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege (STRIDE) methodology [1]. We will use the results of our security assessment to propose an optimal migration scenario of Smart Grid ICS elements to the computing cloud. This is the first published STRIDE methodology ever specifically applied to Smart Grid.

This paper consists of five sections. In Sect. 2 we overview the related works. A somewhat simplified Smart Grid ICS solution architecture and its representative subset of components are described in Sect. 3. Section 4 presents the security risk strategy and methodology. Section 5 describes our STRIDE-based threat analysis of the Smart Grid ICS. Section 6 contains the overview of the optimal Smart Grid ICS to cloud migration scenario.

2 Related Works

In general, most studies about cloud computing applications in power systems are from the technical performance perspectives. In reference [2] the authors claim that cloud computing can significantly improve the operation performances of power systems. As modern power systems are becoming complex cyber-physical systems, the cyber security issues of the power grids have drawn increasingly more attention in recent years. Furthermore, they are the target of a new threat profile that utilizes more sophisticated and targeted attacks than ever before [3]. In references [4–7] the authors consider four kinds of cyber-attacks, which have been proven to be non-neglectful threats to modern industrial systems: compromised keys, eavesdropping, (distributed) denial-of-service (DDoS) and man-in-the-middle (MITM) attacks. Also, in [9], authors present developments related to DDoS attack mitigation solutions in the cloud. Reference [10] elaborates that cyber security for the electric sector is a national concern. The authors claim that concern is growing as the power system becomes increasingly complex and reliant on information technology and communications infrastructures. The assessment scale in this paper is defined based on similar assessment scale in [11] for assessing the overall likelihood of threat events being initiated or occurring and resulting in adverse impacts, annotated by the organization. Security risk analysis, or risk assessment, is one of the fundamental components of an organizational risk management process as described in [12].

Risk assessments are used to identify, estimate, and prioritize risk to organizational operations (i.e. mission, functions, image, and reputation), organizational assets, individuals, other organizations, and the Nation, resulting from the operation and use of information systems [11]. The most important concepts in risk assessment are threats, vulnerabilities, as well as the level of impact and likelihood of occurrence [11, 13].

Threat modeling allows us to identify and rate the threats associated with a system. It might be implemented using one of the following three approaches: asset centric, software centric, and attacker centric [14]. In this paper focus will be on software centric approach, which involves the design of the system and can be illustrated using software architecture diagrams such as data-flow diagrams (DFD), use case diagrams, or component diagrams. All of the system components and data flows are analyzed in relation to the possible attacks

against them, and specific security controls are identified to mitigate identified threats. Threat modeling in Microsoft’s Security Development Lifecycle (SDL) framework is based on this approach [14] and the use of the Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege (STRIDE) threat modeling methodology [15]. References [16, 17] define four types of cloud computing platforms. Private cloud is operated solely for an organization while community cloud supports a specific community that has shared concerns. Public cloud’s infrastructure is available to the general public or a large industry group and is owned by an organization that is selling cloud services. Cloud computing platform whose infrastructure represents a composition of two or more cloud platforms is called hybrid cloud.

The Several European projects focused on a similar topic, so the authors of the Hybrid Risk Management for Utility Providers (HyRiM) [18] have worked risk assessment in Smart Grid systems, while the authors of the SEcure Cloud computing project for CRITICAL IT infrastructure (SECCRIT) [19] have focused on assessing the security of the Cloud platform.

3 Smart Grid ICS Architecture

The Smart Grid ICS consists of several closely related subsystems.

This architecture, shown in Fig. 1, was formed based on relevant scientific and professional literature [3, 4, 20] and on the authors’ own experience.

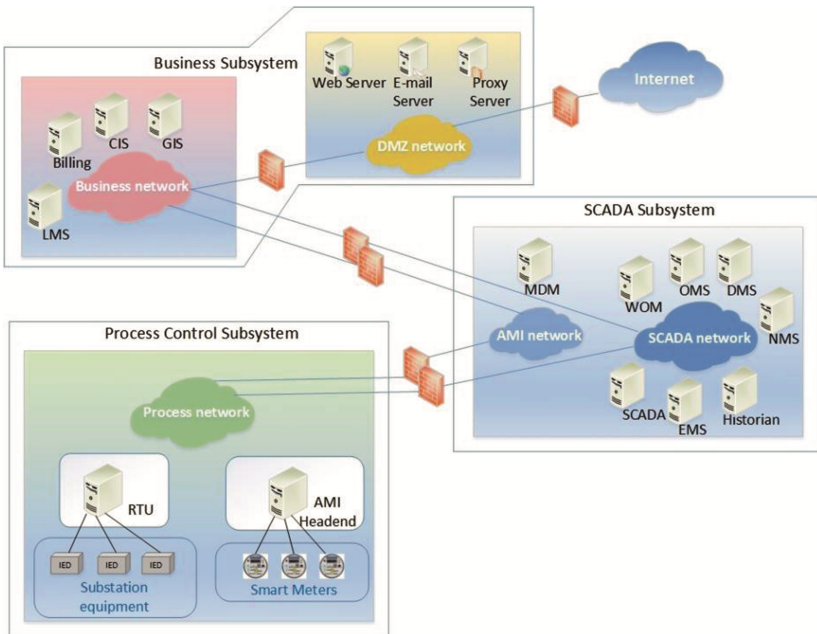


Fig. 1. Simplified ICS architecture in the Smart Grid

The following subsections contain descriptions of each subsystem represented in Fig. 1. Every component of these subsystems will be described in detail in the following subsections. The descriptions will be grouped into three parts and be focused on:

- Process Control Subsystem (i.e. process network),
- Supervisory Control and Data Acquisition (SCADA) subsystem (i.e. SCADA network),
- Business subsystem (i.e. business and DMZ network).

Our analysis will focus on the SCADA subsystem, whose migration to the computing cloud is technically feasible, but feared by most system operators. In accordance with that decision, we will overview the elements of that subsystem in the next sub-section.

The SCADA Subsystem. SCADA subsystem represents the core of Smart Grid IS shown Fig. 1. This section contains the short description of this subsystem's components:

- The Supervisory Control and Data Acquisition System (SCADA) collects data from intelligent electronic devices (IED) inside the smart electricity networks, their transfer to the central authorities of a given system in order to monitor and control system represents the competence of monitoring and control system (SCADA). Remote Terminal Unit (RTU) devices that are located between the central part of the system and equipment in the field perform given collect and send commands to devices that change the state of the system.
- The Outage Management System (OMS) component is used by operators of electric distribution systems to assists in restoration of power. This component handling unplanned outages in the Smart Grid ICS. Closely linked with Work Order Management (WOM) and SCADA module.
- The Network Model Service (NMS) component is responsible for storing and providing access to a network model of the power system. It keeps data in the model which is based on IEC 61970-301 [21] and IEC 61968-11 [22] CIM (Common Information Model) standard and adapted to work with the power calculations. It is only service through which other components can get information about the connectivity of the network.
- The Energy Management System (EMS) performs similar calculations on the transmission and sub-transmission levels, e.g. state estimation, load flow, contingency analysis, and short circuit calculations but on high voltage electric transmission system. It too on the entrance expects a description of the NMS and the current measurement values from SCADA.
- The Distribution Management System (DMS) executes various analytical calculations (topological analysis, load flow, state estimation) on the subsystem for electricity distribution. From NMS takes static model of network, and from SCADA it takes current values of dynamic data.
- The Historian collects and records all changes in the system, incurred as result of the work of other services.
- The Work Order Management (WOM) manages the work orders with which the field workers know exactly what to do and at every moment dispatcher has information about the locations of the field crews, list and statuses of their tasks.

- The Meter Data Management (MDM) works with smart meters and it is usually not connected to the SCADA communication infrastructure. It collects and stores the big data received from the smart meters like information about consumption, and after that it calculates the consumption of individual consumers and transmits that information to the Billing module.

4 Risk Management

Organizations may explicitly define how established priorities and values guide the identification of high-value assets and the potential adverse impacts to organizational stakeholders. If such information is not defined, priorities and values related to identifying targets of threat sources and associated organizational impacts can typically be derived from strategic planning and policies [23].

Risk prioritization is the process of qualification and prioritization of security risks identified during threat modeling. Risk priority is the measure of risk used to show the possible damage to key organizational assets and operations if a threat were to be realized. Risks with the greatest loss and the greatest probability of occurrence are handled first, and risks with lower probability of occurrence and lower loss are handled in descending order.



Fig. 2. Established criteria for impact levels

Security risks are analyzed based on the combined effects of the likelihood of a successful attack and the impact on the identified critical assets. A risk matrix will be used to determine the measure of the security risk. Firstly, the magnitude of impact and likelihood of an exploit are estimated/qualified based on the established criteria for impact and likelihood levels, as given in Figs. 2 and 3, respectively.

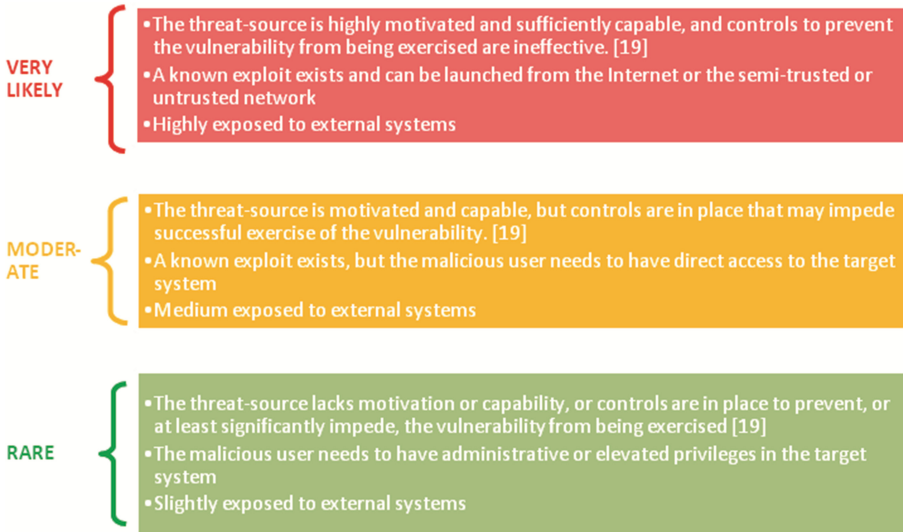


Fig. 3. Established criteria for likelihood levels

Then, the risk rating is determined according to the risk matrix given in Fig. 4.

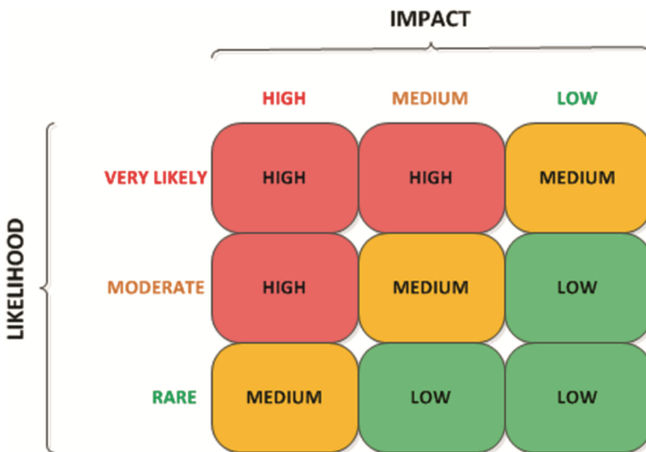


Fig. 4. Risk matrix

Table 1 describes the risk levels shown in the above matrix. Similar risk scale, with its ratings of High, Medium, and Low is described in [23] where necessary actions and corrective measures are proposed based on risk level. There are as well as in [23] represents the degree or level of risk to which an IT system, facility, or procedure might be exposed if a given vulnerability were exercised.

Table 1. Risk matrix description

	Risk Matrix Description
High	<ul style="list-style-type: none"> • If an observation or finding is evaluated as a high risk, an existing system may continue to operate, but a corrective action plan must be put in place as soon as possible.[21] • If can lead to destroying the whole system, losing of human life, damaging to equipment or financial costs. • All components with this degree of risk must be kept in the private cloud.
Medium	<ul style="list-style-type: none"> • If an observation is rated as medium risk, corrective actions are needed and a plan must be developed to incorporate these actions within a reasonable period of time.[21] • All components with this degree of risk should be kept in the private cloud, but if component operates with large volumes of data and it need intermittent access to high computing power it can be moved to community/public cloud.
Low	<ul style="list-style-type: none"> • If an observation is described as low risk, the system's DAA must determine whether corrective actions are still required or decide to accept the risk.[21] • All components with this degree of risk should be moved to community/public cloud.

5 Risk Analysis of Smart Grid ICS Components

Table 2 represents the summarized results of the threat analysis of the identified Smart Grid ICS components from the aspect of the STRIDE **threats (T)**. The impact for each Smart Grid component is assessed and graded with one of the following levels: **Low (L)**, **Medium (M)** and **High (H)**. Levels are numbered according to established criteria for impact in Fig. 2.

The likelihood is determined with one of the following levels: **Very Likely (V)**, **Moderate (M)** and **Rare (R)**. Similarly to impact levels, these levels are numbered according to established criteria for likelihood in Fig. 3.

Finally, the risk (**R**) is determined according to the risk matrix given in Fig. 4.

Table 2. Analysis of SCADA

Supervisory Control and Data Acquisition System		I	L	R
S	The SCADA contains sensitive data, because its inputs are measurements from PCS, and whole system can be compromised by malware users with high privileges. It can cause catastrophic consequences such as damage to infrastructures and financial damage for company. This subsystem is slightly exposed to external systems and internet, but the threat-source is highly motivated.	H	M	H
T	If the value of the data set out of the nominal the consequences can be disastrous. It cause bad operation which can cause a power outages and loss of human lives. This subsystem is slightly exposed to external systems and internet, but the threat-source is highly motivated.	H	M	H
R	Similarly to PCS repudiation of actions in SCADA is a problem, because these actions can lead to catastrophic consequences. This subsystem is slightly exposed to external systems and internet, but the threat-source is motivated.	M	M	M
I	Violation of confidentiality in this component represents a less risky scenario compared to integrity and availability. It can cause disclosure non critical data such as measurements in filed, but these information are not interested for potential attackers. This subsystem is slightly exposed to external systems and internet. The threat-source lacks motivation.	L	R	L
D	In real time systems data delay and data loss is unacceptable. Violation availability of SCADAs data can cause potentially catastrophic consequences such as unusable of the whole system. This subsystem is slightly exposed to external systems and internet, but the threat-source is highly motivated.	H	M	H
E	If unprivileged user gains privileged access such as administrator in SCADA subsystem it can destroy the whole system because he became part of the trusted system. It is very dangerous. This subsystem is slightly exposed to external systems and internet, but the threat-source is highly motivated.	H	M	H

6 Smart Grid ICS Migration to the Cloud

In this section we present a migration scenario of Smart Grid ICS components to a hybrid cloud based on the security analysis presented in the previous sections and summarized in tables in the section “Risk Analysis of Smart Grid ICS components”. CPU processing power usage, cost of setting up and management of different delivery cloud models, and other elements are not factored into this research.

The Hybrid type of the cloud is as an excellent solution for this migration. Hybrid cloud is a composition of private and public/community cloud infrastructures that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds) [16]. Private cloud is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise [17].

The Fig. 5 represents proposed migration scenario Smart Grid components on the cloud.

Based on Table 1, as a general rule of thumb we argue that components should be deployed in the private cloud if their violation can lead to destroying the whole system, losing of human life, damaging to equipment or financial costs and the degree of risk is

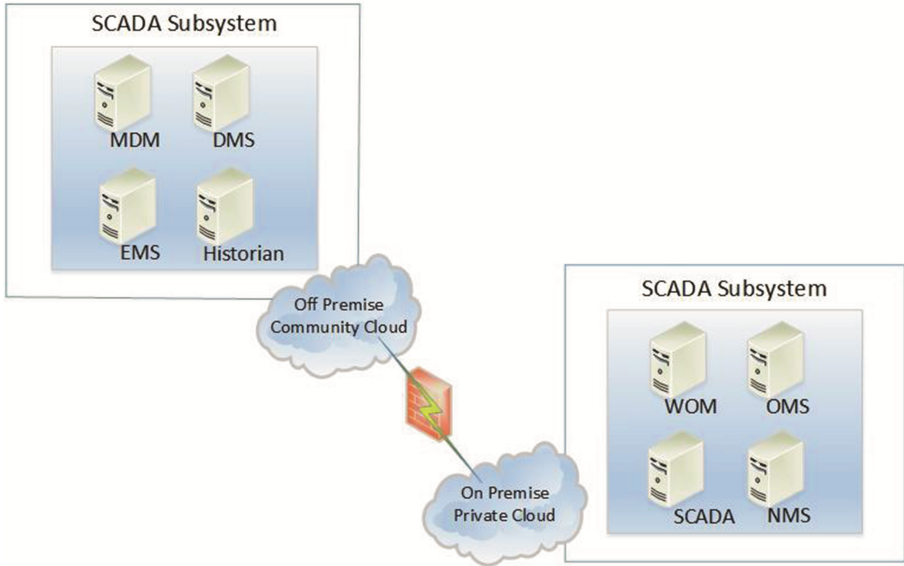


Fig. 5. Secure Smart Grid on a hybrid cloud

high (according to the risk matrix given in Fig. 4). Otherwise, components should be moved to the community cloud. According to the Tables 2, 3, 4, 5, 6, 7 and 8 defined in the previous section, security risk of SCADA, NMS, OMS and WOM is high, and they are located in the private cloud. Data confidentiality is not critical for these components, but integrity and availability represents a serious concern.

Table 3. Analysis of OMS

T	Outage Management System	I	L	R
S	If this component compromise by malware user with high privileged it can cause ignoring of alarm, violation of integrity and other activities that may results damage to equipment or financial costs. This subsystem is medium exposed to external systems and internet. The threat-source is highly motivated.	H	M	H
T	The bad data in OMS make a not valid incidents and it can cause loss of human lives.This subsystem is medium exposed to external systems and internet. The threat-source is highly motivated.	H	M	H
R	Repudiation of creating invalid and unwanted incidents is a big problem, because these actions can lead to human lives and financial losses. This subsystem is medium exposed to external systems and internet. The threat-source is motivated.	M	R	L
I	Violation of confidentiality doesn't affect functioning of the system and doesn't represent a major security risk. Information in OMS are not critical. This subsystem is slightly exposed to external systems and internet. The threat-source lacks motivation.	L	R	L
D	How quickly electro distribution will repair the defect depends on data availability. It can cause financial losses because the response on outage must be in a timely manner. This subsystem is medium exposed to external systems and internet. The threat-source is motivated.	H	M	H
E	The Illegally extension of privileges in OMS can cause creating unnecessary, invalid, deleting of incidents and other unwanted operations that may results damage to equipment, financial costs or loss of human life. This subsystem is medium exposed to external systems and internet. The threat-source is motivated.	H	M	H

Table 4. Analysis of DMS and EMS

T	Energy/Distribution Management System	I	L	R
S	Using another user’s credentials can lead to unwanted calculation performing. If that user has high privilege, it can be dangerous. These subsystems are medium exposed to external systems and internet.The threat-source is highly motivated.	M	M	M
T	Modification of data in the topology analysis and other calculations such as state estimation, load flow, short circuit calculation and other, can significantly affect the final results and system operations and could cause failure in these real time operations. DMS and EMS also uses measurements from SCADA, which integrity is very important. These subsystems are medium exposed to external systems and internet.The threat-source is highly motivated.	M	M	M
R	Each method inside EMS/DMS component has importance in functioning of the whole system, and in providing valid results. Repudiation of their execution is unacceptable.This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	M	R	L
I	Violation of confidentiality represents a less risky scenario in regard to integrity and availability because DMS and EMSdataaren’t interesting in read only mode.This subsystem is medium exposed to external systems and internet.The threat-source is lacks motivated.	L	R	L
D	Practice has shown that the availability of SCADA is more critical.Violation of EMS availability can cause bad operations, but compared to SCADA, it doesn’t have same importance.Similarly to EMS, availability of DMS is important, but not as in SCADA. These subsystems are medium exposed to external systems and internet.The threat-source is highly motivated.	M	M	M
E	If unprivileged user gains high privileges in EMS/DMSunnecessary functions and other unwanted operations could be performed. These subsystems are medium exposed to external systems and internet.The threat-source is highly motivated.	M	M	M

Table 5. Analysis of Historian

T	Historian	I	L	R
S	Malware user with high privileged access on this component can cause violation integrity, availability and confidentiality of this component. It can cause fines and penalties imposed by regulatory bodies or government agencies.This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	M	M	M
T	If the recorded data is not correct analysis will not be and that can cause loss of money and penalties imposed by regulatory bodies or government agencies.This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	M	M	M
R	Repudiation of some actions inside this component isn’t important as repudiation actions in components which are closely associated with it. This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	L	M	L
I	These data include information about state of equipment in the field and about of many other actions. On the basis of their analysis is possible to reconstruct the work of the entire system.This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	M	M	M
D	The Historical data can’t be written during of the unavailability of this service. Similarly to violation of integrity, incompletely of these data can cause loss of money and penalties imposed by regulatory bodies or government agencies. This subsystem is medium exposed to external systems and internet.The threat-source islacks motivated.	M	R	L
E	If unprivileged user gains privileged access on this component it can modify these data and cause violation integrity or delete all data. Such as in previous STRIDE threats it can cause fines and penalties imposed by regulatory bodies or government agencies. This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	M	M	M

Table 6. Analysis of NMS

T	Network Model Service	I	L	R
S	Malware user with administrator’s privileges can cause dangerous problems. It can lead to bad decision in SCADA and it can cause catastrophic consequences such as damage to infrastructures and financial damage for company.This subsystem is medium exposed to external systems and internet.The threat-source is highly motivated.	H	M	H
T	This module contains important information which are necessary for the SCADA and their violation can causelead to bad decisions in the SCADA or other modules. This subsystem is medium exposed to external systems and internet.The threat-source is highly motivated.	H	M	H
R	Based on the above-mentioned possible consequences repudiation of actions in this component is unacceptable because the person responsible for these action must be known.This subsystem is medium exposed to external systems and internet.The threat-source is lack motivated.	M	R	L
I	Information about power network such as connectivity is considered as sensitive information and it may not be revealed. These data are highly regulated by standards.This subsystem is medium exposed to external systems and internet.The threat-source is highly motivated.	M	M	M
D	Similarly to integrity, availability is very important, because violation of availability of this module can cause problem in functioning of SCADA. This threat might cause unusable of whole system.This subsystem is medium exposed to external systems and internet.The threat-source is highly motivated.	H	M	H
E	The Illegally extension of privileges in EMS can cause violation of integrity or delete all data of power network which represent input in system.This threat might cause destroying and unusable of whole system.This subsystem is medium exposed to external systems and internet.The threat-source is highly motivated.	H	M	H

Table 7. Analysis of WOM

T	Work Order Management	I	L	R
S	Using another user’s credentials can lead violate integrity which is very important in this components because may cause the loss of human life (of people in the field) because crews perform instructions from WOM plans. If WOM results are invalid, human’s lives are threatened.This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	H	M	H
T	Violation of integrity may cause the loss of human life such as mentioned above.This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	H	M	H
R	Incorrect data in this component can lead to human losses, and based of that repudiation of actions which associated with making bad data is unacceptable. This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	M	M	M
I	WOM doesn’t contains personal information. These information are important for people in the field.This subsystem is medium exposed to external systems and internet.The threat-source is lacks motivated.	L	R	L
D	Unavailability this component can affects the speed of defects elimination. Financial expenses depend on this speed.This subsystem is medium exposed to external systems and internet.The threat-source is lacks motivated.	M	R	L
E	The Illegally extension of privileges in WOM can cause violation of integrity, and based of that this threat is very dangerous.This subsystem is medium exposed to external systems and internet.The threat-source is motivated.	H	M	H

Table 8. Analysis of MDM

Meter Data Management		I	L	R
S	Using another user’s credentials can lead to unwanted actions. If that user has high privilege as administrator, it can cause invalid bills and dissatisfaction of clients, and it can lead to significant dissatisfaction of clients and company's reputation damage. This subsystem is medium exposed to external systems and internet.The threat-source is highly motivated	M	M	M
T	This data are input in Billing component and their modification can cause can cause problems during billing, e.g. invalid bills for power consumption. Dissatisfaction of clients and company's reputation damage may be results of this.This subsystem is medium exposed to external systems and internet.The threat-source is highly motivated	M	M	M
R	Repudiation in this component is important, but not such as in previously analyzed components.This subsystem is medium exposed to external systems and internet.The threat-source is motivated	L	M	L
I	Based on values from smart meters can be seen which the devices were energized at a given time interval. In this way we violate privacy of end users in Smart Grid and disclosure some information about them.This subsystem is medium exposed to external systems and internet.The threat-source is highly motivated	H	M	H
D	Violation of availability can cause loss of measurements from smart meters, but this problem can be resolved by repeating operations because it isn't real time operation.This subsystem is medium exposed to external systems and internet.The threat-source is lack motivated	M	R	L
E	This threat can cause violation of integrity, availability and confidentiality of this component.This subsystem is medium exposed to external systems and internet.The threat-source is highly motivated	M	M	M

On the other hand, EMS, DMS and Historical components are located in the community cloud. Even though the confidentiality of MDM is critical, it is suggested that MDM is deployed in community cloud, as it operates with large volumes of data and requires high computing power.

The proposed hybrid cloud configuration of Smart Grids has significant benefits, primarily lower infrastructure (e.g. smaller datacenters cost less) and maintenance costs (e.g. smaller IT departments) for companies. Benefits of the public cloud are even more significant, however the level of data security and privacy is guaranteed by the cloud operator. The increased confidentiality level can be reached only if we suppose that the cloud operator employs the best security experts and tools, which the companies in the electric power sector might not be able to do.

7 Conclusion

The goal of this paper was to identify the common elements of a Smart Grid ICS, perform their security assessment and based on that propose a migration scenario to a hybrid computing cloud. A key requirement we had in mind while creating the proposed architecture was to maintain the existing level of information system security. The Smart Grid ICS components were identified by analyzing the relevant scientific and professional literature, as well as based on the authors’ own experience gained via their participation in multiple international SCADA, DMS, EMS and OMS projects.

A risk assessment of the identified software elements was performed by following Microsoft’s Spoofing, Tampering, Repudiation, Information Disclosure, Denial of

Service, Elevation of Privilege (STRIDE) methodology. Such as mentioned in introduction section this is the first published STRIDE methodology ever specifically applied to Smart Grid. Based on the results of the risk assessment, an optimal Smart Grid ICS cloud migration scenario was proposed.

Our analysis is focused on the SCADA subsystem, whose migration to the computing cloud is technically feasible, but feared by most system operators.

In the proposed hybrid cloud-based architecture, the components whose violation can lead to destroying the whole system, losing of human life, damaging to equipment or financial costs are deployed in the private cloud. Otherwise, Smart Grid ICS components are deployed in the community cloud.

As a future work, the authors intend to introduce other measures of the Smart Grid ICS, e.g. factoring in the cost of the necessary computing and storage capacities, the cost of IT departments maintaining the data centers, as well as compliance with relevant security standards and specifications. Also, as a future work we plan to focus on STRIDE analysis of the business and process subsystem.

References

1. Microsoft MSDN documentation, the STRIDE Threat Model. [https://msdn.microsoft.com/en-us/library/ee823878\(v=cs.20\).aspx](https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx)
2. Cao, Z., Lin, J., Wan, C., Song, Y., Zhang, Y., Wang, X.: Optimal cloud computing resource allocation for demand side management. *IEEE Trans. Smart Grid* **8**(4), 1943–1955 (2017)
3. Knapp, E.D.: *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Elsevier (2011)
4. Knapp, E.D., Samani, R.: *Applied Cyber Security and the Smart Grid: Implementing Security Controls into the Modern Power Infrastructure*. Elsevier, Amsterdam (2013)
5. Luo, F., Zhao, J., Dong, Z.Y., Chen, Y., Xu, Y., Zhang, X., Wong, K.P.: Cloud based information infrastructure for next-generation power grid: conception, architecture, and applications. *IEEE Trans. Smart Grid* **7**(4), 1896–1912 (2015)
6. Li, X., Liang, X., Lu, R., Shen, X., Lin, X., Zhu, H.: Securing Smart Grid: cyber attacks, countermeasures, and challenges. *IEEE Commun. Mag.* **50**(8), 38–45 (2012)
7. Aloul, F., Al-Ali, A.R., Al-Daku, R., Al-Mardini, M., El-Hajj, W.: Smart Grid security: threats, vulnerabilities and solutions. *Int. J. Smart Grid Clean Energy* **1**(1), 1–6 (2012)
8. Somani, G., Gaur, M.S., Sanghi, D., Conti, M., Buyya, R.: DDoS attacks in cloud computing: issues, taxonomy, and future directions, 31 March 2017
9. Smart Grid and Cyber Security for Energy Assurance. National Association of State Energy Officials (NASEO), November 2011
10. NIST Special Publication 800-30 Revision 1: Guide for Conducting Risk Assessments
11. NIST SP 800-39: Managing Information Security Risk
12. Whitman, M.E., Mattord, H.J.: *Principles of Information Security*. Course Technology, Boston (2011)
13. Souza, R.D.: 3 Approaches to Threat Modeling, 15 April 2016
14. Mockel, C., Abdallah, A.E.: Threat modeling approaches and tools for securing architectural designs of an e-banking application. In: 2010 Sixth International Conference on Information Assurance and Security (IAS), pp. 149–154. IEEE (2010)
15. Burns, S.F.: GIAC Security Essentials Certification (GSEC) Practical Assignment, Version 1.4c, Threat Modeling: A Process To Ensure Application Security, 5 January 2005

16. The NIST Definition of Cloud Computing. National Institute of Standards and Technology (NIST), Information Technology Laboratory, Version 15 (2009)
17. NIST SP 500-293: US Government Cloud Computing Technology Roadmap Volume I, High-Priority Requirements to Further USG Agency Cloud Computing Adoption. National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899, October 2014
18. Hybrid Risk Management for Utility Providers (HyRiM), EU Project Number 608090
19. SECure Cloud computing project for CRITICAL IT infrastructure (SECCRIT), 01 January 2013–31 December 2015
20. NISTIR 7628: Guidelines for Smart Grid Cyber Security: Vol. 1, Smart Grid Cyber Security Strategy, Architecture, and High-Level Requirements. National Institute of Standards and Technology (NIST) (2010)
21. IEC 61970-301:2013: Energy management system application program interface (EMS-API) - Part 301: Common Information Model (CIM) Base. IEC, Edition 5.0, December 2013
22. IEC 61968-11:2013: Application Integration at Electric Utilities - System Interfaces for Distribution Management - Part 11: Common Information Model (CIM) Extensions for Distribution. IEC, Edition 2.0, March 2013
23. NIST SP 800-30: Risk Management Guide for Information Technology Systems

Cyber Attacks in Industrial Control and Cyber-Physical Systems

Stealthy Deception Attacks Against SCADA Systems

Amit Kleinmann¹(✉), Ori Amichay¹, Avishai Wool¹, David Tenenbaum²,
Ofar Bar², and Leonid Lev²

¹ Cryptography and Network Security Lab, School of Electrical Engineering,
Tel-Aviv University, 6997801 Ramat Aviv, Israel

a.b.kleinmann@gmail.com, oriamich@gmail.com, yash@acm.org

² Israel Electric Corporation, 1 Netiv Ha'Or, 3100001 Haifa, Israel
{david.tenenbaum,br.ofer,leonid.lev}@iec.co.il

Abstract. SCADA protocols for Industrial Control Systems (ICS) are vulnerable to network attacks such as session hijacking. Hence, research focuses on network anomaly detection based on meta-data (message sizes, timing, command sequence), or on the state values of the physical process. In this work we present a class of semantic network-based attacks against SCADA systems that are undetectable by the above mentioned anomaly detection. After hijacking the communication channels between the Human Machine Interface (HMI) and Programmable Logic Controllers (PLCs), our attacks cause the HMI to present a fake view of the industrial process, deceiving the human operator into taking manual actions. Our most advanced attack also manipulates the messages generated by the operator's actions, reversing their semantic meaning while causing the HMI to present a view that is consistent with the attempted human actions. The attacks are totally stealthy because the message sizes and timing, the command sequences, and the data values of the ICS's state all remain legitimate.

We implemented and tested several attack scenarios in the test lab of our local electric company, against a real HMI and real PLCs, separated by a commercial-grade firewall. We developed a real-time security assessment tool, that can simultaneously manipulate the communication to multiple PLCs and cause the HMI to display a coherent system-wide fake view. Our tool is configured with message-manipulating rules written in an ICS Attack Markup Language (IAML) we designed. Our semantic attacks all successfully fooled the operator and brought the system to states of blackout and possible equipment damage.

Keywords: SCADA · Stealthy deception attacks · IDS · NIDS · ICS

1 Introduction

Industrial Control Systems (ICS) are used for monitoring and controlling numerous industrial systems and processes. Supervisory Control And Data Acquisition

(SCADA) system is a type of ICS that typically comprises of different stations distributed over large geographical areas. Anomaly-based intrusion detection approaches are based on “the belief that an intruder’s behavior will be noticeably different from that of a legitimate user” [33]. The main types of anomaly detection approaches that are applied to SCADA systems [3, 4, 10] are: Network-aware detection in which the anomaly detection models only consider network and OS-level events; Protocol-aware detection in which modeling the normal traffic relies on deep-packet-inspection and considers the SCADA control protocol’s meta-data (message sizes, timing, argument addresses, command sequence); and Process-aware approaches which are based on process invariants, mathematical relationships among physical properties of the process controlled by the PLCs.

1.1 Contributions

In this work we present a class of semantic network-based attacks against SCADA systems, that are undetectable by either protocol-aware or process-aware anomaly detection. After hijacking the communication channels between the HMI and PLCs, our attacks manipulate the traffic so as to cause the HMI to present a fake view of the industrial process, thus deceiving the human operator into taking inappropriate and damaging manual actions. Our most advanced attack also manipulates the messages generated by the operator’s actions, reversing the semantic meaning of commands (‘Close’ becomes ‘Open’ and vice-versa) while causing the HMI to present a view that is consistent with the attempted human actions—thus inducing real damage on the cyber-physical system.

Our attacks are totally stealthy to SCADA-aware anomaly detectors since the message sizes and timing, and also the command sequences (including the command arguments) are all 100% legitimate in every way. Furthermore, our attacks are undetectable even by process-aware anomaly detection, since the observed data values of the ICS’s state are completely legitimate: they appear as natural fault conditions that the SCADA system is designed for, and the human operator is trained to handle. Even the operator’s manual command sequences are the expected actions when responding to a natural fault.

We implemented and tested several attack scenarios in the test lab of our local electric company, against a real HMI and real PLCs, separated by a commercial-grade firewall. To do so we developed a real-time security assessment tool, that can simultaneously manipulate the Modbus communication between the HMI and multiple PLCs, and cause the HMI to display a coherent system-wide fake view.

Our tool is configured with message-manipulating rules written in an ICS Attack Markup Language (IAML) we designed. Our tool is near-stateless—it only replaces message contents, without injecting, deleting, extending, or shortening messages. In one scenario we even used the tool in a “half-duplex” mode, wherein it only manipulates the HMI-to-PLC queries, while the PLC-to-HMI responses are unaltered. Despite these self-imposed limitations, our multi-stage semantic attacks all successfully fooled the operator and brought the system to states of blackout and possible equipment damage.

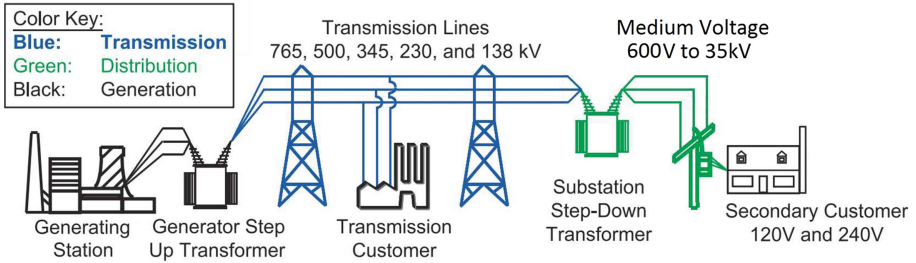


Fig. 1. Basic structure of the electric system (following [1])

The rest of this paper is organized as follows: Sect. 2 explains some fundamentals and assumptions. Section 3 describes the security assessment tool. In Sect. 4 we present three attacks we tested with their impact on the HMI and the PLCs. Section 5 succinctly reviews related work and in Sect. 6, we suggest possible countermeasures and conclusions.

2 Preliminaries

2.1 Electrical Distribution

An electricity supply chain is usually divided into three subsystems: generation, transmission, and distribution, as depicted in Fig. 1. Electricity is transported along high voltage transmission lines (the transmission network) over long distances, from generation sites to major distribution points. The transmission lines are connected to distribution substations. At a distribution substation, a substation transformer takes the incoming transmission-level voltage (138 to 765 kV) and steps it down to several distribution primary circuits (“medium-voltage” circuits, 600 V to 35 kV), which fan out from the substation. Close to each end user, a distribution transformer takes the medium-voltage and steps it further down to a low-voltage secondary circuit (commonly 120/240 V). In this paper we focus on the distribution subsystem, between the substation and distribution transformers—which is precisely the subsystem impacted during the Ukrainian cyber-attacks [27, 28].

For improved reliability, distribution circuits are often provided with “tie switches” to other circuits which are normally open (i.e., disconnected). If a fault occurs on one of the circuits, the tie switches can be closed (connected) to let electricity flow into the faulted circuit, and to allow some portion of the service to be restored. The tie switches can be operated either manually, or automatically from the SCADA system interface. These switches, also called switchgears, may be simple open-air isolator switches or may be gas-insulated.

2.2 Adversary Model

Our underlying threat model is loosely based on the Dolev-Yao threat model [13]: The adversary may overhear and intercept all traffic regardless of its source and

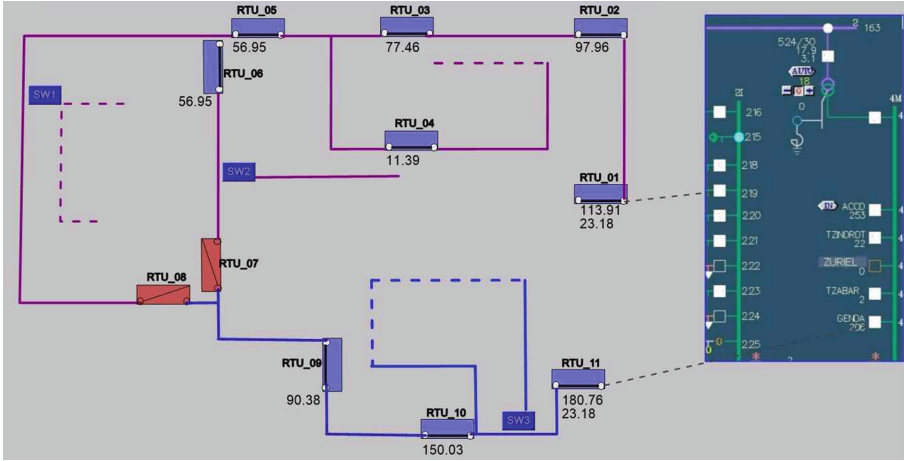


Fig. 2. A screenshot of the HMI panel. We see the substation is on the right, and 2 radial distribution lines: the top line (purple, via RTU 01–06) and the bottom line (blue, via RTU 11,10, 09). RTU 07 and 08 control the switchgears that tie the two lines: both are shown as open (disconnected). The numbers below each RTU show the current (113.91A and 180.76A at the line heads, dropping to 11.39A at RTU 04). At the two line heads (RTU 01, 11) the HMI also shows the voltage (23.18 kV) displayed. (Color figure online)

destination. More precisely, we assume the adversary has a Man-In-The-Middle (MITM) position between the HMI and *all* the PLCs. The adversary can inject, delete, and delay arbitrary packets with any source and destination addresses on the communication channels it controls. Consequently, the adversary can also replay previously overheard messages, or manipulate messages in transit. The objective of the adversary is to manipulate the SCADA network to achieve an impact on the physical world.

We further assume that the adversary has in-depth knowledge of the architecture of the SCADA network and the various PLCs as well as sufficient knowledge of the physical process and the means to manipulate it via the SCADA system. Thus the adversary has the ability to fabricate messages that would result in real-world physical damage.

In our experiments we implemented a somewhat weaker type of attacker: Our attack tool is near-stateless and does not track or modify the TCP sequence numbers. Hence our attacks do not inject fabricated messages or drop legitimate ones: our attack tool only modifies the contents of pre-existing messages. Despite this self imposed restriction, our attacks are all successful, and undetectable by suggested anomaly-detection systems.

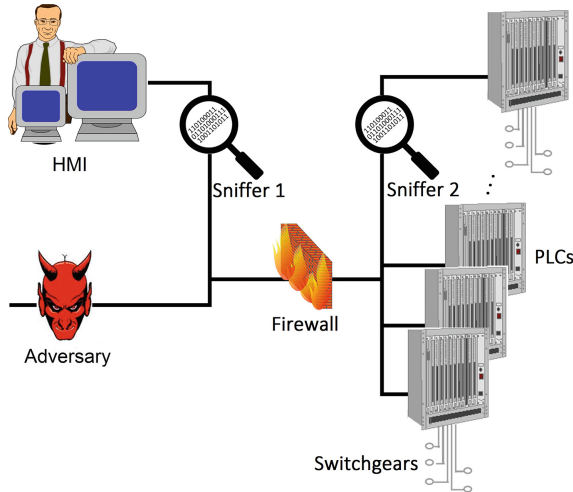


Fig. 3. Network diagram of the test lab

2.3 The Test Lab

The electric company test lab consists of 11 Unitronics V130 PLCs, controlled by a Cimplicity version 9.5 HMI running on a Microsoft Windows 2012 server. The HMI and the PLCs are connected to separate VLANs and separated by Check Point 4000 appliance R77.30 firewall.

The test lab emulates a substation with 2 radial distribution lines, that are interconnected by tie switches. Moreover, along each line there are several additional switchgears, see Fig. 2. In total there are 11 PLCs, each controlling a switchgear: each PLC reports back to the HMI the voltage and current flowing through it, and the switchgear state (open/closed), and accepts commands to open or close the switchgear.

The distribution system controlled by the PLCs is simulated by a separate PLC (the S-PLC) - which the switchgear PLCs interface with. Whenever a switch attempts to read a sensor value (e.g., current), the S-PLC provides the required value. The sensor values reported by the S-PLC are based on measurements taken at real switchgears that are deployed at a certain radial circuit of our local electric company's network.

The switchgears that are controlled by RTU_07 and RTU_08 in Fig. 2 are the tie switches and are initially *disconnected*. The initial state of all the other switchgears is *connected*.

The HMI runs two separate polling threads that monitor the eleven PLCs using the Modbus protocol. The polling threads repeatedly send the same three read-requests to get register values from each one of the PLCs. The PLCs all run the same control logic and expose the same Modbus memory layout: in particular the current value is stored in register #130 and the voltage value is stored in register #131.

In this environment, and indeed in the electric company’s real HMI, switchgears are only operated manually from the HMI: If the operator observes a fault, such as current and voltage dropping to zero, she can open or close the switchgears by clicking on the HMI screen. Such operator actions are realized by corresponding Modbus ‘write’ packets that are sent from the HMI to the proper PLC. Note that in the test lab, the S-PLC reacts to such write events by updating all the subsequent current and voltage values that will be reported to all the relevant PLCs to be consistent with the system state following the operator action.

3 The Attack Tool

3.1 Gaining Network Access

There are many ways for an attacker with network access to place itself in a MITM position. In our attack tool we chose to implement a well known ARP poisoning attack (cf. [2]). Using ARP poisoning obviously makes the attack tool detectable to standard low level Network Intrusion Detection Systems (NIDS). However, our focus is on semantic SCADA-aware anomaly detection, so we assume the attacker is able to bypass the NIDS somehow.

In order to record the attacks’ progress, we placed 2 traffic sniffers, one on each of the VLANs (Sniffer 1 and Sniffer 2 in Fig. 3).

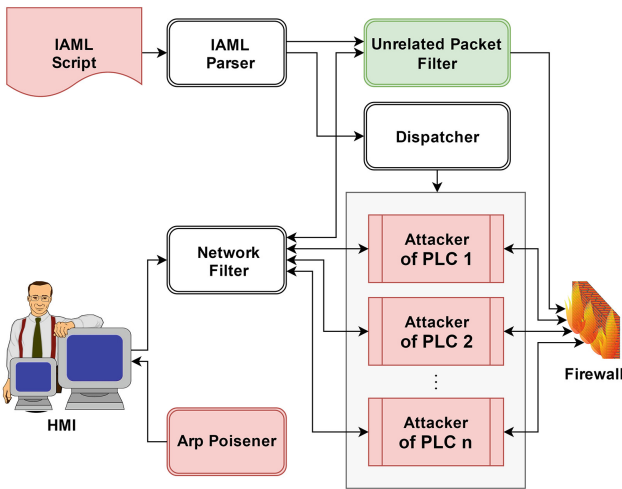


Fig. 4. The architecture of the attack tool

The architecture of the attack tool is depicted in Fig. 4. It comprises of: an *Arp poisoner* that crafts and sends the spoofed-ARP messages to the HMI, an *Unrelated Packet Filter (UPF)* that just forwards these packets (untouched) to

their original destination, a *Dispatcher* that creates a *PLC attacker* (composed of 2 threads, sharing state, per each direction of the traffic) per each PLC that needs to be attacked. An *IAML parser* that accepts an IAML script, parses it and transfers the needed parameters to the *Dispatcher* and the *UPF*, and a *Network filter* that filters proper packets from the HMI VLAN and sends them to the appropriate *PLC attacker*. The Network filter is based on the Pcap.Net .NET wrapper for WinPcap. In our full technical report [21] we describe the attack tool in more details.

3.2 Near-Stateless Manipulation of Modbus

Modifying the message length in the Modbus stream is a relatively “noisy” attack action. SCADA-aware anomaly detection that has even minimal Modbus understanding can flag messages with unusual lengths. Further, injecting messages into a hijacked connection is also detectable by either a network-aware approach (if the message timing is unusual) or by a protocol-aware approach (if the function code or arguments are unusual). Therefore, to demonstrate the power of our attacks we elected to make our attack tool stateless at the transport layer: it does not track or modify the TCP sequence numbers at all. Note that this introduces a significant limitation on the attacker: e.g., it precludes replacing a Modbus “read” query by a “write” query (which is the usual way to implement commands)—simply because a “write” message is longer, due to the additional written-values payload.

However our tool is not totally stateless: As we shall see, we wish to modify the values reported to the HMI in PLC responses of selected messages. Importantly, the Modbus response messages do not carry the read register addresses; they carry only the read data values, while the register addresses are present in the HMI’s query message. Thus when the attack tool matches a query message of interest, it places that query’s Modbus Transaction ID (TID) into a state variable; when the corresponding response message, with the same TID, is matched on the same connection, the attack tool modifies its content.

4 Semantic Attacks Against Electric Distribution SCADA

4.1 Zero Values Deception Attack

Our first deception scenario is illustrated in Fig. 5. The attacker invokes the attack twice (first for 18 seconds between 151.8–169.8s, and second for 14s, between 359.53–373.53s). During each one of these attacks the attacker simultaneously changes the values of the registers (#130 and #131) that hold the current and the voltage reported by six PLCs to be zeros. The victim PLCs are those controlling the top line (recall Fig. 2): RTU 01–06. The bottom graph of Fig. 5 shows the true current and voltage values as recorded by Sniffer 2 on the PLC VLAN. The top graph shows the values of the same registers as recorded by Sniffer 1 and observed at the HMI.

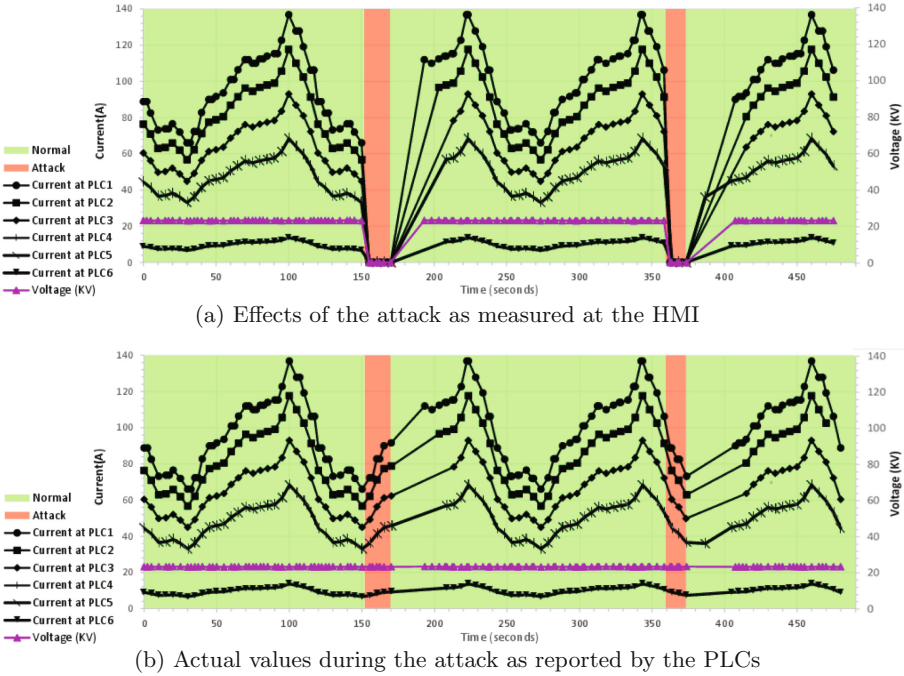


Fig. 5. The zero values deception attack against an ICS of an electric distribution system. The attack sends zero values for the current and voltage reported simultaneously by RTU_01 - RTU_06 (denoted here as PLC1-PLC6 respectively), causing the HMI to present a view consistent with a fault on the top line.

This attack causes the HMI to display a view in which RTU_01 - RTU_06 all show zero current - as in Fig. 6. This view is consistent with a natural fault on the top line – and causes the operator to implement unneeded remediation steps, that are expensive and possibly harmful. Note that the attack is super-stealthy: SCADA-aware anomaly detection is blind to such an attack, since the attack mimics a natural fault, which is a planned-for scenario and does not, in itself, signify an attack.

4.2 A Multi-stage Attack

Our main attack is more elaborate, and aims to interfere with the operator’s reaction to a fake fault. In this attack scenario we implemented a stealthy multi-stage deception attack, see Fig. 7. The attack has two main stages: the first stage is the Zero values deception attack described in the previous section. Looking at the HMI panel (depicted at Fig. 6) the operator is deceived to believe that the top radial circuit is disconnected. This motivates the operator to start disconnecting and reconnecting switchgears according to the operating procedures. This is where the second stage of the attack comes into action. In this stage,

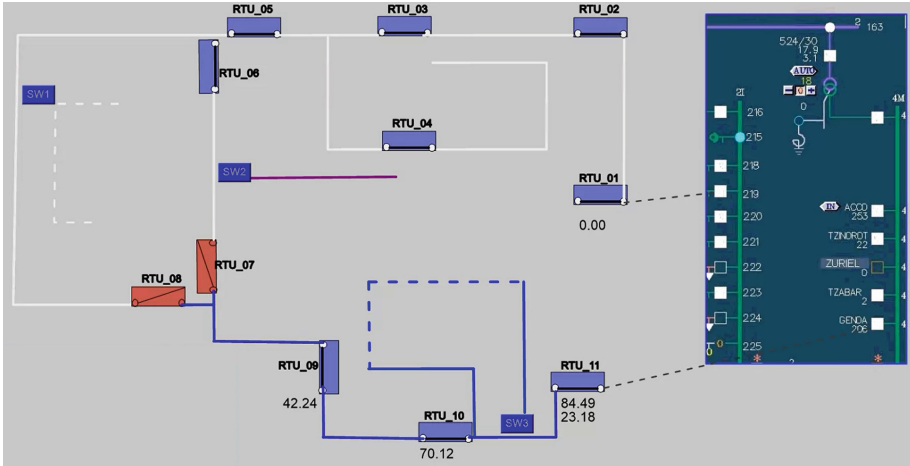


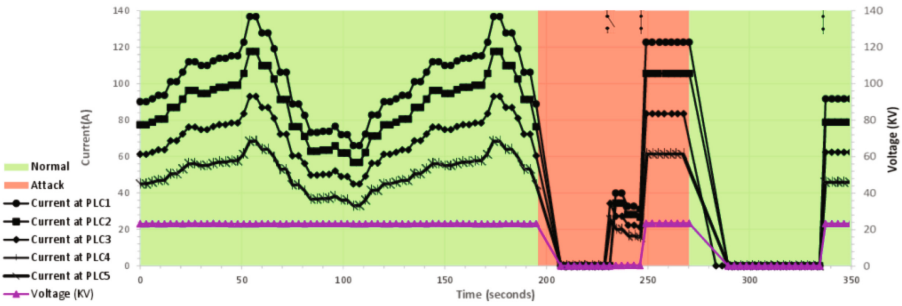
Fig. 6. The HMI panel showing a disconnection in the top radial line. Note the voltage of 0.0kV at RTU_01, the missing current value at RTUs 02–06, and the white color of the line. In such a scenario the operator would typically open the switchgear at RTU 01 and close the ties at RTU 07, 08 to attempt to supply the top line from the bottom line.

whenever the operator issues a switchgear open/close command, the attack tool replaces it with the *opposite command*.

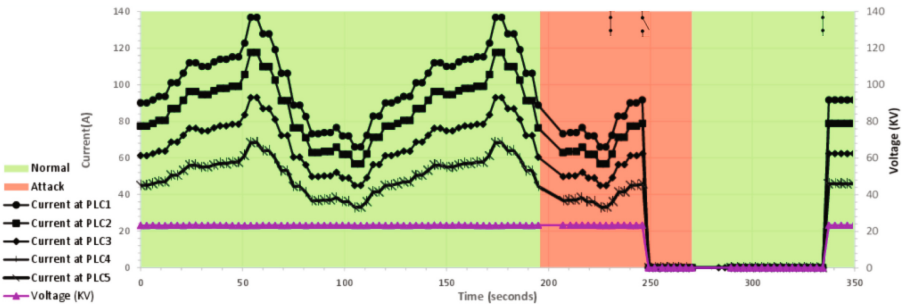
The bottom graph of Fig. 7 shows the actual values (of current and voltage) recorded at the different PLCs, and the top graph shows the manipulated values observed at the HMI. The two graphs also indicate the open/close actions by small icons: the top graph shows the intended operator actions at the HMI (*Open* at time 231.3 and *Close* at time 246.6), and the bottom graph shows the icons for the reversed action received and executed by the PLCs. Note that the attack also fakes current and voltage values that are consistent with the intended operator actions: after the *Open* command the attack starts returning current values computed as “nominal – actual”, where “nominal” is a fixed per PLC constant; this shows the operator that the trouble shooting has some effect. Then after the *Close* command (at time 246.6) the attack tool reports “nominal” current values on all the PLCs—while in fact the circuit is disconnected and customers are experiencing a blackout. Again, note how stealthy the attack is: all Modbus messages are on schedule, using normal functions and arguments, with designed-for, semantically reasonable, data values.

4.3 A Half-Duplex Attack

Recall that the zero-values attack of Sect. 4.1 required the attacker to modify values in responses from the PLC to the HMI. In this section we describe an attack that is equivalent to the zero-values attack, except it requires a simpler setup by the attacker: Here the attacker acts as a MITM only on the traffic



(a) Effects of the attack as measured at the HMI



(b) Actual values during the attack as reported by the PLCs

Fig. 7. The multi-stage attack – the attack starts at time 195.8 with a zero-values attack. The second stage executing the “opposite operation” attack starts at 231.3 and ends at 270.4, with operator commands at times 231.3 and 246.6. The icons at the top edge of the graphs indicate the *Open* and *Close* commands to the switchgears.

that flows from the HMI to all the PLCs while the PLC-to-HMI responses are unaltered. Hence the attacker is unable to directly set data values to zero in PLC responses. To achieve an attack in this scenario, query messages (from the HMI to the PLC) reading the current and voltage values are modified to access registers whose addresses are shifted by 1. As a result, during the attack, the shifted register-values sent by the PLCs are interpreted at the HMI as the values of the corresponding preceding registers—the HMI interprets the voltage value as the current, and the content of register #132 (which is 0) is interpreted as voltage-value, again fooling the operator into deducing that a fault occurred. We omit the graphs.

Note that this attack can be detected by a protocol-aware anomaly detection [18,24] as long as the detector is located at the network segment where the PLCs reside, since the attacker modifies the accessed register addresses, which creates unknown symbols in the GW model [18].

4.4 The ICS Attack Markup Language

We defined a new formalism for specifying a concrete execution of an attack, an ICS Attack Markup Language which we called an IAML. IAML enables planning and implementation of a multi-stage, multi-PLC, simultaneous attacks on an ICS without any a priori programming knowledge. It uses a modular approach, where a module represents an attack that changes a certain type of packet under certain conditions. Modules can be linked together to create multi-stage attacks. The relationships among modules are specified through the conditions and changes of local and global stages. IAML is accompanied by a library of predicates, which function as a vocabulary to describe the properties of attack modules and stages. In our full technical report [21] we describe the details of IAML and include complete usage example.

5 Related Work

5.1 Attacks Against ICS

Digital attacks that cause physical destruction of equipment do occur [19]. Most recently, cyber-attacks on SCADA systems controlling electrical distribution have caused wide-spread blackouts in Ukraine [27,28]. Perhaps most well known is the attack on an Iranian nuclear facility in 2010 (Stuxnet) to sabotage centrifuges at a uranium enrichment plant. The Stuxnet malware [15,26] worked by changing centrifuge operating parameters in a pattern that damaged the equipment – while sending normal status messages to the HMI. In 2014, the German Federal Office for Information Security announced a cyber attack at an unnamed German steel mill, where hackers manipulated and disrupted control systems to such a degree that a blast furnace could not be properly shut down, resulting in “massive”-though unspecified-damage [12].

Byres et al. [7] describe the application of the attack tree methodology to SCADA communication systems based on the common Modbus protocol stack. The authors identify eleven possible attacker goals with their respective technical difficulty, possible severity of impact and likelihood of detection. In particular they noted that an attacker can perform a Man-In-The-Middle (MITM) attack between a PLC and HMI and “feed” the HMI with misleading data, allegedly coming from the PLC – which is what we implemented.

In 2009 Fovino et al. [16] showed that malware was able to disrupt or even seize control of vital sensors and actuators. Semantic attack scenarios on ICSs are described by [17,31] for a system with a pipe in which high pressure steam or fluid flows. The pressure is regulated by two valves. An attacker capable of sending packets to the PLCs can force one valve to complete closure, and force the other to open. Each of these ICS commands is perfectly legal when considered individually, however when sent in an abnormal order they can cause a ‘water hammer’ and bring the system to a critical state.

5.2 Anomaly Detection in ICS

Surveys of techniques related to learning and detection of anomalies in critical control systems can be found in [3, 4, 10]. While most of the current commercial network intrusion detection systems (NIDS) are signature-based, i.e., they recognize an attack when it matches a previously defined signature, anomaly-based NIDS “are based on the belief that an intruder’s behavior will be noticeably different from that of a legitimate user” [33]. All anomaly detection approaches below do not distinguish between malicious events and faulty events — and none of them is able to detect our attacks.

Network–Aware Detection. Basic anomaly detection models for SCADA systems only consider network and OS-level events. Yang et al. [38] used an Auto Associative Kernel Regression (AAKR) model coupled with the Statistical Probability Ratio Test (SPRT) and applied them on a SCADA system. The model used numerous indicators representing network traffic and hardware-operating statistics to predict the ‘normal’ behavior.

Barbosa et al. [5, 6] analyzed SCADA traces they collected at two different water treatment and distribution facilities. They concluded that SCADA traffic presents remarkably regular time series, due to the fact that the majority of the traffic sources generate data in a periodical fashion. They selected only the high energy frequencies for the anomaly detection phase.

Protocol–Aware Detection. More advanced anomaly detection systems rely on deep-packet-inspection and consider the ICS control protocol’s meta-data, modeling command sequences, and argument addresses.

Model-based anomaly detection for SCADA systems, and specifically for Modbus traffic, was introduced by Cheung et al. [11]. They designed a multi-algorithm intrusion detection appliance for Modbus/TCP with pattern anomaly recognition, Bayesian analysis of TCP headers and stateful protocol monitoring, complemented with customized Snort rules [35]. In subsequent work, Valdes et al. [37] incorporated adaptive statistical learning methods into the system to detect for communication patterns among hosts and traffic patterns.

Goldenberg and Wool [18] developed a model-based approach (the GW model) for Network anomaly detection based on the normal traffic pattern in Modbus SCADA networks using a Deterministic Finite Automata (DFA) to represent the cyclic traffic. The SCADA messages are modeled both in isolation and also by their sequence order. Subsequently, Kleinmann et al. [22, 23] demonstrated that a similar methodology is successful also in SCADA systems running the Siemens S7 protocol.

Caselli et al. [9] proposed a methodology to model sequences of SCADA protocol messages as Discrete Time Markov Chains (DTMCs). Based on data from three different Dutch utilities the authors found that only 35%–75% of the possible transitions in the DTMC were observed. This strengthens the observations of [5, 18, 22] of a substantial sequentiality in the SCADA communications.

However, unlike [18,22] they did not observe clear cyclic message patterns. The authors hypothesized that the difficulties in finding clear sequences is due to the presence of several threads in the HMI's operating system that multiplex requests on the same TCP stream.

Kleinmann et al. [23–25] introduced a modeling approach for multiplexed SCADA streams, using *Statechart DFAs*: the *Statechart* includes multiple DFAs, one per cyclic pattern. Each DFA is built using the learning stage of the GW model. Following this model, incoming traffic is de-multiplexed into sub-channels and sent to the respective DFAs.

Process-Aware Detection. These anomaly detection methods are based on process invariants: mathematical relationships among physical properties of the process controlled by the PLCs. Several publications [8,30,36] explain that an IDS that models only the protocol meta-data (commands and arguments) is not sufficient, since attacks can be mounted using legitimate control commands, but with attacker-selected data values. To combat such attacks they suggest modeling both the physical process and the continuous control function. Based on measurements of the state of the process, the models predict the control's response and its effect on the state, and flag deviations from the predicted state.

Fovino et al. [17] use detailed knowledge of the industrial process' control to generate a system virtual image representing the PLCs of a monitored system. The virtual image is updated using a periodic active synchronization procedure and via a feed generated by the intrusion detection system (i.e., known intrusion signatures).

Hadziosmanovic et al. [20] used the logs generated by the control application running on the HMI to detect anomalous patterns of user actions on process control application.

Lin et al. [29] combine system knowledge of both the control network (extracting control commands from SCADA network packets) and the physical infrastructure in power grid (obtaining measurements from sensors in substations) to help IDS to estimate execution consequences of control commands, thus to reveal attacker's malicious intentions. The authors claimed that their semantic analysis provides reliable detection of malicious commands with a small amount of analysis time.

Erez et al. [14] developed an anomaly detection system that detects irregular changes in SCADA control registers' values. The system is based on an automatic classifier that identifies several classes of PLC registers (Sensor, Counter and Constant registers). Parameterized behavior models were built for each class. In its learning phase, the system instantiates the model for each register. During the enforcement phase the system detects deviations from the model.

Mo et al. [32] as well as Pasqualetti et al. [34] investigated the detection and prevention of deception and replay attacks. They concluded that certain types of attacks are undetectable by using their attack models.

6 Conclusions and Counter Measures

This work presented a class of semantic network-based attacks against SCADA systems which are undetectable by both protocol-aware and process-aware anomaly detection. After hijacking the communication channels between the HMI and PLCs, our attacks cause the HMI to present a fake view of the industrial process, deceiving the human operator into taking manual actions. Our most advanced attack also manipulates the operator's commands reversing their semantic meaning while causing the HMI to present a view that is consistent with the attempted operator directions. The attacks are totally stealthy since the message sizes and timing, the command sequences, and the data values of the ICS's state all remain legitimate. They appear as natural fault conditions that the SCADA system is designed for, and the human operator is trained to handle.

We implemented and tested several attack scenarios in the realistic test lab of our local electric company. We developed a real-time security assessment tool, that can simultaneously manipulate the communication to multiple PLCs, and cause the HMI to display a coherent system-wide fake view. Our tool is configured with a new IAML language we designed. Our multi-stage semantic attacks all successfully fooled the operator and brought the system to states of blackout and possible equipment damage.

We argue that current intrusion detection and anomaly detection systems are fundamentally unable to detect the stealthy deception attacks we described. In fact, once the attacker is positioned as MITM, the traffic at both the HMI and the PLC sides looks perfectly normal—because it is perfectly normal. In our opinion the only real countermeasure against such attacks is to secure the communication channel via cryptographic means. E.g., by adding data integrity protections such as digital signatures or message authentication codes to block the attacker's ability to modify packets. Nevertheless, we believe that ongoing research into anomaly detection for ICS is still very valuable: with such systems in place, the attacker is restricted to *only* mount super-stealthy deception attacks like ours, and cannot mount simpler and more direct attacks without risk of detection.

References


1. Final report on the August 14, 2003 blackout in the United States and Canada: Causes and recommendations. U.S.-Canada Power System Outage Task Force, U.S. Secretary of Energy and Minister of Natural Resources Canada, April 2004
2. Abad, C.L., Bonilla, R.I.: An analysis on the schemes for detecting and preventing ARP cache poisoning attacks. In: 27th International Conference on Distributed Computing Systems Workshops, ICDCSW 2007, pp. 60–60. IEEE (2007)
3. Alcaraz, C., Cazorla, L., Fernandez, G.: Context-awareness using anomaly-based detectors for smart grid domains. In: Lopez, J., Ray, I., Crispo, B. (eds.) CRiSIS 2014. LNCS, vol. 8924, pp. 17–34. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17127-2_2

4. Atassi, A., Elhajj, I.H., Chehab, A., Kayssi, A.: The state of the art in intrusion prevention and detection. In: *Intrusion Detection for SCADA Systems*, Chap. 9, pp. 211–230. Auerbach Publications, January 2014
5. Barbosa, R., Sadre, R., Pras, A.: A first look into SCADA network traffic. In: *IEEE Network Operations and Management Symposium (NOMS)*, pp. 518–521, April 2012
6. Barbosa, R., Sadre, R., Pras, A.: Towards periodicity based anomaly detection in SCADA networks. In: *17th IEEE Emerging Technologies Factory Automation (ETFA)*, pp. 1–4, September 2012
7. Byres, E.J., Franz, M., Miller, D.: The use of attack trees in assessing vulnerabilities in SCADA systems. In: *International Infrastructure Survivability Workshop (2004)*
8. Cárdenas, A.A., Amin, S., Lin, Z.S., Huang, Y.L., Huang, C.Y., Sastry, S.: Attacks against process control systems: risk assessment, detection, and response. In: *6th ACM Symposium on Information, Computer and Communications Security*, pp. 355–366. ACM (2011)
9. Caselli, M., Zambon, E., Kargl, F.: Sequence-aware intrusion detection in industrial control systems. In: *1st ACM Workshop on Cyber-Physical System Security*, New York, NY, USA, pp. 13–24 (2015). <http://doi.acm.org/10.1145/2732198.2732200>
10. Chen, C.M., Hsiao, H.W., Yang, P.Y., Ou, Y.H.: Defending malicious attacks in cyber physical systems. In: *2013 IEEE 1st International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*, pp. 13–18, August 2013
11. Cheung, S., Dutertre, B., Fong, M., Lindqvist, U., Skinner, K., Valdes, A.: Using model-based intrusion detection for SCADA networks. In: *SCADA Security Scientific Symposium*, pp. 127–134 (2007)
12. De Maizière, T.: Die Lage der IT-Sicherheit in Deutschland 2014. The German Federal Office for Information Security (2014). https://www.google.co.il/url?sa=t&rect=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwigs8_B1enXAhVSFuwKHQm3Ba8QFggmMAA&url=https%3A%2F%2Fwww.bmi.bund.de%2FSharedDocs%2Fdownloads%2FDE%2Fpublikationen%2F2014%2Fbsi-lagebericht-it-sicherheit.pdf%3F__blob%3DpublicationFile&usg=AOvVaw2deYBrgkWuS45W4MBrUldL
13. Dolev, D., Yao, A.C.: On the security of public key protocols. Technical report, Stanford, CA, USA (1981)
14. Erez, N., Wool, A.: Control variable classification, modeling and anomaly detection in Modbus/TCP SCADA systems. *Int. J. Crit. Infrastruct. Prot.* **10**, 59–70 (2015)
15. Falliere, N., Murchu, L., Chien, E.: W32.Stuxnet dossier. White paper, Symantec Corporation, Security Response (2011)
16. Fovino, I.N., Carcano, A., Masera, M., Trombetta, A.: An experimental investigation of malware attacks on SCADA systems. *Int. J. Crit. Infrastruct. Prot.* **2**(4), 139–145 (2009). <http://www.sciencedirect.com/science/article/pii/S1874548209000419>
17. Fovino, I., Carcano, A., De Lacheze Murel, T., Trombetta, A., Masera, M.: Modbus/DNP3 state-based intrusion detection system. In: *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pp. 729–736. IEEE (2010)
18. Goldenberg, N., Wool, A.: Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *Int. J. Crit. Infrastruct. Prot.* **6**(2), 63–75 (2013). <http://www.sciencedirect.com/science/article/pii/S1874548213000243>
19. Gorman, S.: Electricity grid in U.S. penetrated by spies. *Wall Street J.* A1 (2009). <http://www.wsj.com/articles/SB123914805204099085>

20. Hadziosmanovic, D., Bolzoni, D., Hartel, P.H., Etalle, S.: MELISSA: towards automated detection of undesirable user actions in critical infrastructures. In: European Conference on Computer Network Defense, EC2ND, Gothenburg, Sweden, USA, pp. 41–48, September 2011
21. Kleinmann, A., Amichay, O., Wool, A., Tenenbaum, D., Bar, O., Lev, L.: Stealthy deception attacks against SCADA systems. [arXiv:1706.09303](https://arxiv.org/abs/1706.09303) [cs.CR], June 2017
22. Kleinmann, A., Wool, A.: Accurate modeling of the siemens S7 SCADA protocol for intrusion detection and digital forensic. *JDFSL* **9**(2), 37–50 (2014). <http://ojs.jdfsl.org/index.php/jdfsl/article/view/262>
23. Kleinmann, A., Wool, A.: A statechart-based anomaly detection model for multi-threaded SCADA systems. In: Rome, E., Theocharidou, M., Wolthusen, S. (eds.) *CRITIS 2015*. LNCS, vol. 9578, pp. 132–144. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33331-1_11
24. Kleinmann, A., Wool, A.: Automatic construction of statechart-based anomaly detection models for multi-threaded SCADA via spectral analysis. In: 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy, CPS-SPC 2016, pp. 1–12. ACM, New York (2016). <http://doi.acm.org/10.1145/2994487.2994490>
25. Kleinmann, A., Wool, A.: Automatic construction of statechart-based anomaly detection models for multi-threaded industrial control systems. *ACM Trans. Intell. Syst. Technol. (TIST)* **8**(4), 55 (2017)
26. Langner, R.: Stuxnet: dissecting a cyberwarfare weapon. *IEEE Secur. Priv.* **9**(3), 49–51 (2011)
27. Lee, R.M., Assante, M.J., Conway, T.: Analysis of the cyber attack on the Ukrainian power grid. Technical report, SANS E-ISAC, 18 March 2016. https://ics.sans.org/media/E-SAC_SANS_Ukraine_DUC_5.pdf
28. Liang, G., Weller, S.R., Zhao, J., Luo, F., Dong, Z.Y.: The 2015 Ukraine blackout: implications for false data injection attacks. *IEEE Trans. Power Syst.* **32**(4), 3317–3318 (2017)
29. Lin, H., Slagell, A., Kalbarczyk, Z., Sauer, P.W., Iyer, R.K.: Semantic security analysis of SCADA networks to detect malicious control commands in power grids. In: First ACM Workshop on Smart Energy Grid Security, pp. 29–34. ACM (2013)
30. Liu, Y., Ning, P., Reiter, M.K.: False data injection attacks against state estimation in electric power grids. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **14**(1), 13 (2011)
31. Marsh, R.T.: Critical foundations: protecting America’s infrastructures - the report of the president’s commission on critical infrastructure protection. Technical report, President’s Commission on Critical Infrastructure Protection, October 1997
32. Mo, Y., Kim, T.H.J., Brancik, K., Dickinson, D., Lee, H., Perrig, A., Sinopoli, B.: Cyber-physical security of a smart grid infrastructure. *Proc. IEEE* **100**(1), 195–209 (2012)
33. Mukherjee, B., Heberlein, L.T., Levitt, K.N.: Network intrusion detection. *IEEE Netw.* **8**(3), 26–41 (1994)
34. Pasqualetti, F., Dörfler, F., Bullo, F.: Attack detection and identification in cyber-physical systems. *IEEE Trans. Autom. Control* **58**(11), 2715–2729 (2013)
35. Roesch, M.: Snort - lightweight intrusion detection for networks. In: 13th USENIX Conference on System Administration, LISA 1999, pp. 229–238. USENIX Association, Berkeley (1999). <http://dl.acm.org/citation.cfm?id=1039834.1039864>
36. Urbina, D.I., Giraldo, J.A., Cardenas, A.A., Tippenhauer, N.O., Valente, J., Faisal, M., Ruths, J., Candell, R., Sandberg, H.: Limiting the impact of stealthy attacks on industrial control systems. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1092–1105. ACM (2016)

37. Valdes, A., Cheung, S.: Communication pattern anomaly detection in process control systems. In: IEEE Conference on Technologies for Homeland Security (HST), pp. 22–29 (2009)
38. Yang, D., Usynin, A., Hines, J.: Anomaly-based intrusion detection for SCADA systems. In: 5th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies, pp. 12–16 (2006)

On Ladder Logic Bombs in Industrial Control Systems

Naman Govil¹, Anand Agrawal²(✉), and Nils Ole Tippenhauer² 

¹ IIIT Hyderabad, Hyderabad, India
naman.govil@research.iiit.ac.in

² Information Systems Technology and Design Pillar,
Singapore University of Technology and Design,
8 Somapah Road, Singapore 487372, Singapore
{agrawal_anand,nils_tippenhauer}@sutd.edu.sg

Abstract. In industrial control systems, devices such as Programmable Logic Controllers (PLCs) are commonly used to directly interact with sensors and actuators, and perform local automatic control. PLCs run software on two different layers: (a) firmware (i.e. the OS) and (b) control logic (processing sensor readings to determine control actions).

In this work, we discuss *ladder logic bombs*, i.e. malware written in ladder logic (or one of the other IEC 61131-3-compatible languages). Such malware would be inserted by an attacker into existing control logic on a PLC, and either persistently change the behavior, or wait for specific trigger signals to activate malicious behavior. For example, the LLB could replace legitimate sensor readings with manipulated values. We see the concept of LLBs as a generalization of attacks such as the Stuxnet attack. We introduce LLBs on an abstract level, and then demonstrate several designs based on real PLC devices in our lab. In particular, we also focus on *stealthy LLBs*, i.e. LLBs that are hard to detect by human operators manually validating the program running in PLCs.

1 Introduction

Industrial Control Systems (ICS) are computer systems that typically control physical processes that relate to power, water, gas, manufacturing and other critical infrastructure. ICS and Supervisory Control and Data Acquisition (SCADA) systems rely on local programmable logic controllers (PLCs) to interface with sensors and actuators. While PLC devices are available from a range of manufacturers, they are all commonly programmed with the same set of programming languages based on IEC 61131-3. In particular, the IEC 61131-3 standard [7] contains *ladder logic*, *functional block diagram*, and *sequential text* as different languages that are used together to define logic to run on the PLCs. The logic is then interpreted by the firmware running on the PLCs. Modern PLCs provide security mechanisms to allow only legitimate (e.g., signed) firmware to be uploaded. In contrast, logic running on the PLCs can typically be altered by anyone with network or local USB access to the PLC. This setting is the main

difference to malware scenarios in traditional corporate IT environments, where the injection of attacker code is usually significantly harder.

Recently, the security of Cyber Physical Systems (CPS) and related systems has gained a lot of attention [4, 13, 21–23]. In particular, CPS such as critical infrastructure including power grids, nuclear power plants, and chemical plants are threatened. In CPS, physical-layer interactions between components have to be considered as potential attack vectors, in addition to the conventional network-based attacks.

In this work, we introduce *ladder logic bombs* (LLBs), i.e. malware written in ladder logic (or one of the other IEC 61131-3-compatible languages). LLBs consist of logic that is intended to disrupt the normal operations of a PLC by either persistently changing the behavior, or by waiting for specific trigger signals to activate malicious behavior. In particular, the LLBs could lay dormant and hence hidden for a very long time until a specific trigger is observed. Once activated, the LLB could replace legitimate sensor readings that are being reported by the PLC to the SCADA system with manipulated values. We introduce LLBs by classifying their purpose and action, and demonstrate several constructions based on real PLC devices in our lab.

We implemented and tested our attacks on a real-world ICS (the SWaT testbed, see Sect. 4). In particular, we focused on *stealthy LLBs*, i.e. LLBs that are hard to detect by human operators manually validating the program running in PLCs. We provide a classification of logic based attacks, such as the ones performed by Stuxnet [5].

We summarize our contributions as following:

- We analyzed firmware updates on the target platform to detect vulnerabilities.
- We identify the issue of logic manipulations on PLCs, and introduce the concept of *ladder logic bombs* (LLBs).
- We present a range of LLB prototypes, in particular ones that attempt to hide from manual logic code inspection.

The structure of this work is as follows: In Sect. 2, we introduce CPS systems, PLCs, and IEC 61131-3 in general. We propose our Ladder Logic Bomb concept in Sect. 3, and present example implementations in Sect. 4. Related work is summarized in Sect. 6. We conclude the paper in Sect. 7.

2 Background

In this section, we will introduce some of the salient properties of industrial control system (ICS) networks that we have found so far. In addition, we will briefly introduce Ladder Logic programming language and the tools necessary to interact with such PLCs.

2.1 ICS

In the context of this work, we consider ICS that are used to supervise and control system like public infrastructure (water, power), manufacturing lines, or

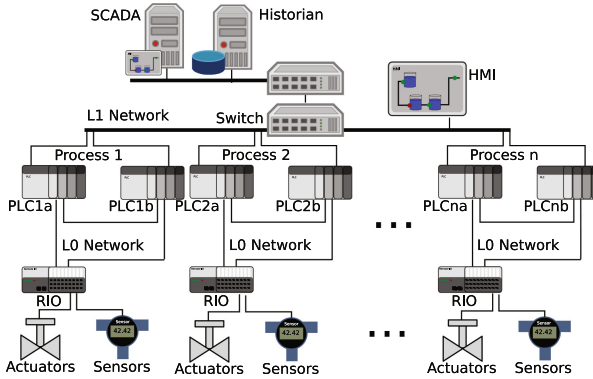


Fig. 1. Example local network topology of a plant control network.

public transportation systems. In particular, we assume the system consists of programmable logic controllers, sensors, actuators, and supervisory components such as human-machine interfaces and servers. We focus on single-site systems with local connections, long distance connections would in addition require components such as remote terminal units (see below). All these components are connected through a common network topology.

2.2 Ladder Logic and Studio 5000

A Programmable Logic Controller (PLC) is an industrial computer system that continuously monitors the state of input devices and makes decisions based on a custom program to control the state of output devices. PLCs are widely used in industrial control systems (ICS) for handling sensors and actuators, mainly because of their robust nature and ability to withstand harsh conditions including severe heat, cold, dust, and extreme moisture. Considering their widespread usage and important nature of tasks handled by PLCs, their security against malicious manipulation is critical.

PLC programs are typically written in a special application on a local host (personal computer), and then downloaded by either a direct-connection cable or over a network to the PLC. The program is stored in the PLC in a non-volatile flash memory. While details differ for platforms from alternative vendors, it might be required to enable remote change of control software on the PLC through a physical switch (i.e., *program* mode on ControlLogix devices). We observe that due to convenience, in practical systems PLCs are often kept in that setting to allow easy remote access. In addition, any attacker with physical access is able to change the switch setting easily. For that reason, we assume that remote or local reprogramming access is possible in the remainder of this work.

IEC 61131-3 is an open international standard [7] for PLCs that defines two graphical and one textual programming language standards for PLCs:

- Ladder Logic Diagrams (graphical)
- Functional block Diagram (graphical)
- Structured Text (textual)

The most popular of those languages is Ladder Logic Diagrams. The main intuition behind this Ladder Logic Diagrams is to provide a system-wiring diagram abstraction similar to electro-mechanical relays. Ladder logic is more of a rule-based graphical language implemented by *rungs*, rather than traditional procedural-based language. A rung in the ladder represents a rule. They are called “ladder” diagrams because they resemble a ladder, with two vertical rails (supply power) and as many “rungs” (horizontal lines) as there are control circuits to represent.

“Studio 5000” is a software product of Rockwell Automation that provides an environment to develop a range of elements for a control system, for operational and maintenance use. Its major element is the Studio 5000 Logix Designer application, formerly (RSLogix 5000), software to program Logix5000 controllers.

Another tool called RSLinx is used to establish USB-based communication between PLCs and a host PC running Studio 5000. RSLinx is a Windows based software package to interface with a range of ICS and automation hardware. In this paper, we used Allen-Bradley PLCs (ControlLogix 5571) with Studio 5000 v21.00. It is important to note that for different PLCs, different versions of RSLinx and Studio 5000 have to be used.

3 Ladder Logic Bombs

In this section, we present our proposed concept of *ladder logic bombs*. In particular, we noticed that while changes to the firmware of PLCs are made more difficult by digital signatures, the actual logic that is executed on the PLCs is not protected by such a measure. In addition, the lack of security checks/authentication before downloading new logic onto PLCs is a cause of major concern. An attacker can exploit this by either gaining physical access to the PLCs or over the network, and can download custom (*malicious*) logic onto PLCs which can compromise the system. Next, we discuss potential attack scenarios and goals, which can be achieved through this vulnerability.

3.1 System and Attacker Model

In this work, we assume that the attacker is able to access PLCs in an industrial control system either remotely via the network, or physically. As we will show, commonly such access will allow the attacker to read and modify the programming logic of the PLCs without any authentication. The attacker is assumed to have access to the respective software required to download and upload logic configurations to the PLC (e.g., Studio 5000 for ControlLogix PLCs).

The goals of the attacker can range from achieving a Denial of Service (DoS), to changing the behavior of the PLCs, or to obtain data traces of sensor and



Fig. 2. Ladder Logic Bomb (LLB) classification

control messages processed by the PLC. In order to perform these attacks, the attacker just needs access to the PLC system once, making such attacks all the more dangerous. The attacker could also have sporadic (physical) access to the PLC. For example, the attacker only has access to the PLC once a week (because he is a regular contractor). In these events, the attacker can trigger any behavior changes (i.e. trigger his ladder logic bomb) at a point unrelated to his access time (e.g., to hide correlations to his access).

The system we consider in this setting is very generic and can be described as follows: a PLC in an industrial control system which uses IEC 61131-3 languages for the logic, and can be re-programmed as described above. It is connected to sensors and actuators of a critical process. Operators of the plant configured the logic of the PLC at design time. Though they continuously monitor the status of these PLCs, they seldom need to change the logic configuration of the already operational system. They are also able to manually download the logic to inspect it, if required. Although we will briefly discuss a network-based detection mechanism using an intrusion detection system later, such a solution will not be able to detect changes by a local attacker. For that reason, we do not focus on IDS in this work. In addition, physical layer prevention mechanisms (camera, fences, etc.) are out of scope of this work.

We do not consider an attacker that is able to attack the operator's machines (as it was the case in Stuxnet), or able to manipulate network traffic while it is being transmitted. In particular, if the attacker was able to compromise the operator's machine, then the operator would not be able to verify any code reliably. Such an attacker could be addressed by using a trusted computing platform, which we consider out of scope for this work. The attacker model does also not consider insider attacks (e.g., an attacker who might be regular contractor/employee with authorization to access and modify the PLC logic).

3.2 Bomb Classification

Ladder logic bombs can be classified broadly by two criteria (as shown in Fig. 2). LLBs can be classified according to their activation and triggering. They can either be externally triggered by giving a certain input. Alternatively, they can be triggered by internal logic (system states, specific instructions or data, clock, etc.)

LLBs can also be classified according to the alteration they incur onto the existing PLC system. They can add or remove certain functionality in the existing logic (*modify function*). These bombs can also alter the system values such as system date/time, timezone, wall-clock time, or similar (*modify system*).



Fig. 3. Overview of SWaT testbed (photo source: <http://itrust.sutd.edu.sg>).

Finally, these can also be used for data exfiltration and transmitting crucial system data to a spy node (*transmit information*).

Together, those classifications now describe more specific LLBs. For example, a LLB that turns off a pump at 12 AM would be classified as internally activated function modification LLB.

3.3 Triggering

Here, we describe the different triggering mechanisms that can be used with ladder logic bombs.

Triggering at a particular Input: The bomb could be set off when a pre-determined input is detected. For example, we are targeting a water treatment ICS for our experiments (see Sect. 4.1). The target PLC is receiving inputs about the water level in one of the tanks from its corresponding level sensor. The bomb could be set off when a particular level is reached in the tank.

Triggering Sequence: The bomb could also be triggered when a particular trigger sequence is detected. This would potentially make the bomb more difficult to detect, as none of its effects would be visible until the particular sequence is detected as input. This can be achieved by implementing a finite state machine (FSM) using latches.

Timer: The bomb could also be set off using a timer. This would make the LLB like a real world time bomb, which sets into motion when the timer has finished its count sequence. Using nested TON timers, it is possible to implement count sequences which will last days.

Specific Internal Condition: The bomb could be triggered when a particular internal state is achieved. This particular triggering scheme requires the attacker to have complete knowledge and understanding of the logic on the PLCs. When a particular state variable, for example a fault code, is set, the bomb could be set off and the payload logic is executed.

3.4 Hiding LLBs in PLC Logic

The naïve approach to detect any modifications in the original logic (in our case, the LLBs) would be to download the control logic from PLC devices, and manually inspect them for code changes. In particular, engineers familiar with the plant operations might be able to read through the code and detect malicious changes. While that approach might be feasible for small sites and very simple logic, we will show in the following section that there are several options for the attacker to hide the malicious payload within the logic to make it harder to detect by such manual inspection.

4 Implementation

In this section, we describe in detail the construction of ladder logic bombs and demonstrate how they can be used to disturb the functioning of ICS.

4.1 SWaT Testbed

The experiments were conducted on an industrial control system testbed, called SWaT, located at the Singapore University of Technology and Design. Secure Water Treatment, as depicted in Fig. 3, is a fully functional (scaled down) water treatment plant. SWaT was constructed exclusively as platform for research on cyber physical system security. The water treatment process is partitioned into six stages, starting with raw water in Tank 1 to filtered output water in Tank 6. Each stage is controlled by an independent PLC which determines control actions using data from sensors.

Sensors values and actuator commands are communicated to and from a PLC via a plant network. The system also contains monitors to view and ensure system states are within acceptable operational boundaries. Data from sensors are available for inspection on the Supervisory Control and Data Acquisition (SCADA) workstation and recorded by the Historian for subsequent analysis.

4.2 Attack 1: DoS Using Add on Instructions

The Denial of Service (DoS) is a potential attack goal to inflict (most often financial or reputation) damage on critical systems. In a DoS attack, the attacker temporarily or permanently slows or stops correct operations of a system.

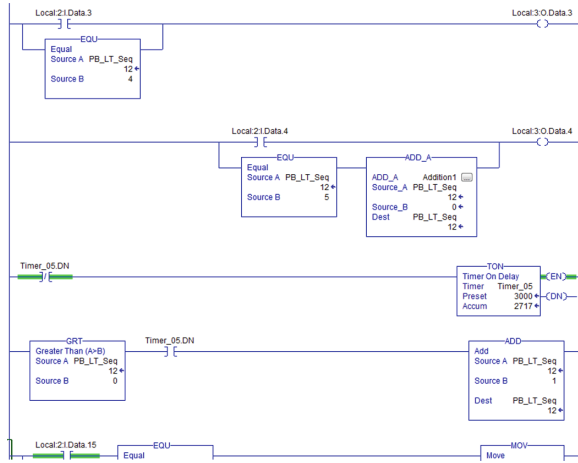


Fig. 4. Malicious Add-On instruction

On the Internet, (distributed) DoS attacks are often achieved by creating massive amounts of traffic that overload communication links or servers. As PLCs control the action of sensor and actuators in the system, their operational availability is often critical [12]. If the PLC is incapable of controlling the actuators, it can have disastrous consequences (e.g., lead to the loss of control of heavy machinery in an automobile assembly plant).

Goal: In this setup, the goal was to launch a DoS attack on one of the PLCs in a water treatment plant.

Construction: This has been achieved by implementing an infinite loop as the bomb payload. The trigger mechanism for this LLB is when a particular input is received. Similar to Stuxnet [5], the trigger check condition lays on top of the actual logic, which always stays on to check if the particular input has been received. As soon as the desired trigger input is received, the LLB springs into action.

Concealment: The actual malicious logic has been hidden inside an Add-On Instruction. A new instruction has been created, which is very similar in its construction to the real *ADD* block, with similar inputs: 2 sources A and B and an output: Destination. It has also been named suitably (*ADD_A*) to disguise well with a real *ADD* block. From the top overview of the ladder logic (which contains many rungs), this looks just like any other *ADD* block on one of the rungs. But inside this add-on instruction, the real bomb (an infinite loop) is defined, and that adversely affects the PLC operation. More details about this can be found in Fig. 4.

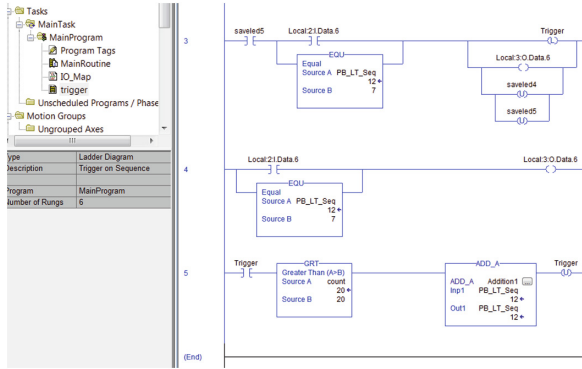


Fig. 5. Inside the exploiting subroutine

4.3 Attack 2: Manipulation of Sensor Data Using Subroutines

Another important function of the PLCs in ICS (in addition to controlling the actuators) is reading data from sensors. That data can be critical information about the process and system. Using the data, it is possible to derive the current state of the process, which is used by the PLC to determine appropriate control actions. Thus, tampering with sensor data can cause systems to fail [11].

Goal: The goal for this attack was to manipulate sensor readings coming from the remote IOs (RIOs in Fig. 1) to the PLC.

Construction: Since this is proof-of-concept, we decided to manipulate the sensor values and increase them by a constant offset (we arbitrarily chose four). As result, the LLB payload is a simple *ADD* block which takes the real sensor values and increases them by four, and stores them back into the same tag. However, a more complex triggering mechanism was used in this attack. In particular, the LLB is triggered when a complete trigger sequence is detected. This has been achieved by implementing a finite state machine using latches (see Fig. 5).

Concealment: For this attack, we also used a different hiding technique. By inspecting the actual logic of the PLC in the water treatment plant, we observed that the logic was calling a large number of subroutines. We assume the subroutines were called that way to maintain good readability of the ladder logic by the maintainers. However, that structure with large number of subroutines can be leveraged by the attacker to hide the LLB. We tested this exploit by hiding a trigger subroutine that gets executed every cycle of the ladder logic (see Fig. 6).

4.4 Attack 3: Data Logging Using FFLs

The attacks discussed above are openly causing damage or malfunctions, and their effects can be observed as soon as triggered. However, there are another



Fig. 6. Overview of the logic with the exploiting subroutine

class of LLBs which can be equally harmful but are harder to detect. In particular, such LLBs could be used for data logging and exporting sensitive information about the system.

Goal: The goal of this attack is to achieve stealthy data logging of sensitive information about the plant.

Construction: The data logging is achieved by using a FIFO buffer which reads data into an array. The *FFL* block has been used for this purpose. As shown in Fig. 7, the *FFL* block stores the tag *PB_LT_Seq* which contains sensitive information about the count sequence used to determine state of the plant. Those values are stored into the *array2* and are converted into *.csv* format and stored on the SD card in the PLC. Staying within our attacker model, an attacker who has sporadic access (physical access to PLCs) to the plant can come in, read these values stored on the SD card. Then, insert this card back into the PLC and leave. The trigger sequence for this could be a simple timer, thus ensuring data logging after ‘x’ days of plant operation.

Concealment: This LLB can again be concealed either inside an Add-On instruction or as a subroutine. It can also be left inside the main logic flow, since this LLB contains just one extra rung, making its manual detection difficult in large and complex code.

4.5 Attack 4: Trigger Major Faults on PLC

We now discuss another attack which is similar in effect to the DoS attack.

Goal: The goal is to trigger major faults on the PLC which causes its processor to halt and which cannot be fixed by a hard reset.

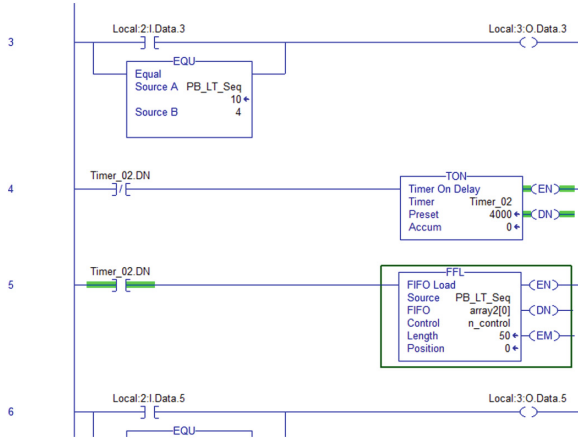


Fig. 7. Data logging in a FIFO buffer

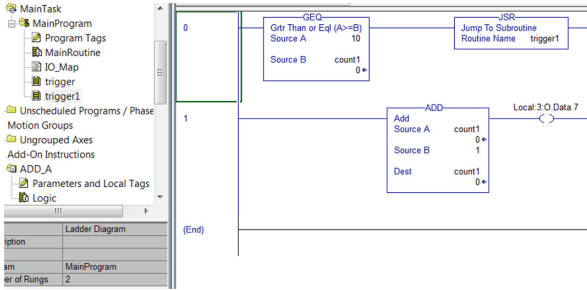


Fig. 8. Stack overflow

Construction: Here we managed to cause a major faults on the PLC. In particular, we used *invalid array subscripts* and a *stack overflow*. The stack overflow was achieved by implementing a recursive subroutine call to itself. This caused the stack storing the return pointer to overflow, halting the process and crashing the PLC (Fig. 8).

Concealment: These LLBs can be concealed within an Add-On instruction or inside a subroutine.

4.6 Analysis of Attacks

Ideally, there would be a metric to measure the *stealthiness* of LLBs, that would indicate how hard different LLBs are to discover. So far, we have not found a good way to measure that property. In the following, we instead use the *relative additional lines of code* (RALOC) to measure the stealthiness. In particular, the increase of lines of code in the logic can also lead to increased memory

consumption at runtime. We observed that there are two types of memory that is used by a ladder logic program: I/O memory and Data & Logic memory. As part of our analysis, we measured the difference (increase) in memory of the original logic when malicious ladder logic bombs were added. It was observed that there was no increase in the I/O memory of the PLC at all, which is primarily because no new inputs/outputs were created to trigger or apply the ladder logic bombs discussed above. The only increase observed was in the data and logic memory, which is also marginal, as depicted in Table 1. Important thing to note is that Table 1 entries are taken at particular attack scenario and the size of Attack 3 (data logging) will depend on the amount of data that is logged. As result, the RALOC metric increases, and the modifications might become more visible.

To mitigate that effect, it is best to save the data on the SD card and then flush the arrays so that they can be re-used if more data needs to be logged.

Table 1. Comparison of attacks performed

Attack	Increase in memory (%)
Attack 1: DoS using AOI	2.60
Attack 2: Manipulate sensor	3.84
Attack 3: Data logging	3.41
Attack 4: Major faults	4.09

5 Countermeasures

In this section, we discuss potential countermeasures against LLB attacks. In particular, we discuss (a) network-based countermeasures, and (b) centralized validation of running code.

In the following, we assume that the countermeasures are retro-fitted into an existing industrial control system. In particular, we assume it is not possible to change the PLCs themselves. If we could change the way logic updates are applied to PLCs, it would trivially be possible to introduce user authentication (e.g. with username/password, or public key-based), or cryptographic signatures for logic updates. The PLC would then only accept the logic code update if the user is successfully authenticated, or the authenticity of the update has been validated.

5.1 Network-Based Countermeasures

If an intrusion detection system (IDS) is already used in the network to monitor traffic for spreading malware or other malicious traffic, then that IDS could potentially be used to identify the specific traffic related to logic updates on PLCs connected to the network. If unauthorized logic updates over the network are observed, an alarm could be raised. A similar IDS is proposed in [6], where

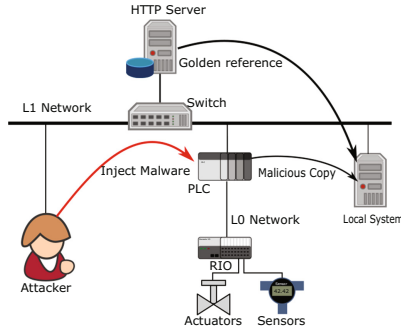


Fig. 9. Centralized logic store based countermeasure

the authors model periodic communication between HMI and PLCs using a deterministic finite automata. The system flags anomalies if a message appears out of position in normal (general) sequence of messages. If the IDS is configured to operate as intrusion prevention system (IPS), the offending traffic could even be dropped in real time.

The problem with this proposal is related to the identification of authorized logic updates. As we cannot change the traffic generated by the respective software, there is no way to embed specific authentication information. Thus, we can only use information such as IP source address (supposedly related to the authorized person), which is not ideal (as it can be spoofed).

5.2 Centralized Logic Store

Our second proposal is based on two components: (a) a centralized logic store (CLS) of the latest version of logic running on all PLCs of the ICS, and (b) a tool to periodically download currently running logic from the PLCs, and to validate that against the “golden” copy from the CLS. An overview of our proposed system can be found in Fig. 9.

Submission of golden samples: All authorized engineers are required to submit the most recent version of logic for each PLC to the CLS when they change the logic running on the PLCs. To do so, they can use a simple application that requires them to identify the respective logic file, the target PLC, and their credentials. That application will then use the credentials to establish an authenticated secure channel to the CLS (e.g. using TLS), and then upload the latest logic version to the CLS (e.g. using HTTP over the established TLS session).

Periodic Logic Validation: We have implemented a python-based tool to manually and periodically validate the logic. The user first exports ladder logic to a *.L5K* file (sequential text) on the local machine using Studio 5000. Next, our tool parses the *.L5K* file and extracts a unique serial number corresponding to the logic. Then, the tool connects to the CLS where the correct golden logic is

searched by using Beautiful Soup parser (BSP). BSP is a python library to parse HTML and XML pages, in our case BSP parse CLS and look for all *.L5K* file followed by our parser which looks for correct golden logic by identifying the unique serial number.

Then, the tool performs a comparison between the logic found on the PLC, and the golden sample. If differences are found, they can be visualized to a human operator using standard functionality provided by tools such as *diff*. The algorithm below summarizes the whole process.

Algorithm 1. CLS based countermeasure

Require: Downloaded malicious PLC logic (.L5K) file
 Establish server connection at specific port
 Parse local .L5K file and fetch serial no.
 GET golden sample from server with serial no.
if diff(local .L5K, golden reference .L5K) == 0 **then**
 Local logic successfully validated
else
 Local logic differs, present diff to user
 User manually inspects code differences
 if User detects attack **then**
 Raise alarm
 else if Local Logic newer **then**
 Update golden sample on CLS if authorized
 else
 Update local logic with golden sample
 end if
end if

6 Related Work

General Threats to ICS. It has been observed over the years that process control systems are vulnerable to various exploits with potentially damaging physical consequences [2, 14, 20].

In [19] Morris and Gao discuss different attacks such as measurement injection, command injection, denial of service, etc., on SCADA control systems which use the MODBUS communication protocol. Much like the rest, this study is again restricted to exploiting the network layer to attack the PLCs. Therefore, it is necessary to analyze control logic vulnerabilities, which can be manifested through malicious logic additions.

Stuxnet. In 2010, Stuxnet [5] caused a radical shift in focus for security of such control systems by demonstrating practical exploitation of the control logic in these devices. This resulted in increasing focus on security aspects of PLCs and their control logic [1, 8, 9]. In [8], Karnouskos discuss Stuxnet and how it managed to deviate the expected behavior of PLC. In [9], Kim discuss the cyber security

issues in nuclear power plants and focused on stuxnet inherited malware attacks on control system, and its impacts in future along with its countermeasures.

Protocol-based attacks. The authors of [1] discuss replay, reconnaissance and authentication by-pass attacks. These attacks can be performed by sending probe requests or by examining the ISO-TSAP conversation and authenticating oneself by generating packets with same hash, in turn, achieving access to PLC logic. All these attacks are focused on exploiting the communication protocols to gain access to PLCs.

In [18], the authors investigate vulnerabilities of industrial PLCs on firmware and network level, leaving out any analysis on logic level exploits. In this work, we provide a consolidated study on logic layer manipulations and provide logic level safeguarding methods, unlike the network based security (e.g., firewall, VPN security and secured layered architecture) methods proposed in majority of the papers above.

Control Logic Manipulation. In [15], the authors propose a PLC malware capable of dynamically generating a payload based on observations of the process taken from inside the control system. The malware first gathers clues about the nature of the process and the layout of physical plant. Dynamic payload is then generated to meet the specific payload goal. However, the authors assume that an attacker must be insider or have prior knowledge of the targeted system. That dependency is worked upon in [16], which proposes a tool to automatically determine semantics of the target PLC, minimizing the need for prerequisite knowledge of target control system. This work however does not go into details of malicious logic construction on ladder logic or any other IEC 61131-3-compatible language and focus mainly on network layer attack.

Countermeasures. In general, attempting to validate the authenticity of the root file system or files/directories is not a new concept. In [10], Kim and Spafford proposed a monitoring tool “Tripwire”. It monitors the Unix based file system and notifies the system administrator in case a corrupted file or alteration is detected. In contrast to Tripwire tool (which uses interchangeable signature subroutines to identify changes in file) our proposed CLS based countermeasure compares the local instance of a file with its authorized one. Another important point to note here is that the Tripwire tool is host based, used for unix based file systems whereas proposed countermeasure is used in respect to PLC logic file (.L5K) extracted from Studio 5000 tool.

We found a number of works focused on development of countermeasure techniques to safeguard PLCs and other components of industrial control systems. In [3], a sequence aware intrusion detection system (S-IDS) is proposed. The IDS focuses on detection certain sequences of events (e.g. sensor readings or control actions) that are harmless on their own, but can lead to unwanted consequences if chained together. Other attack detection methods for PLCs are found in [17, 24]. In [24], the authors propose an approach based on symbolic execution of PLC code along with control model checking to automatically detect the malicious code running on the PLC. In [17], a Trusted Safety Verifier (TSV)

is implemented on a Raspberry PI set-up, placed in between the control system network and the PLC as a bump-in-the-wire to intercepts all the controller code and validate it against all the safety properties defined by process engineer. This requires additional hardware set-up to function. In this paper, we intend to propose countermeasures which can be very easily used with the traditional (existing) industrial control system architecture and have least dependency on PLC internals (construction and interface internals).

7 Conclusion

In this paper, we have introduced the term *ladder logic bombs* to discuss the problem of logic malware for PLCs, such as modifications performed by Stuxnet [5]. Contemporary vulnerabilities study for such devices usually do not include analysis on control logic level, which is an important source of attacks as demonstrated in this work. We analyzed vulnerabilities in the firmware running on PLCs and depicted case studies and attack scenarios in real-time on actual PLCs to inflict damage on industrial control systems. All the tests were conducted on a real world ICS, unlike majority of the theoretical works presented in the literature so far. Finally, a centralized logic store based countermeasure technique was proposed and implemented, that can detect logic level based attacks effectively.

Acknowledgments. This work was supported by SUTD's startup grant SRIS14081.

References

1. Beresford, D.: Exploiting Siemens Simatic S7 PLCs. In: Proceedings of Black Hat USA (2011)
2. Cárdenas, A.A., Amin, S., Sastry, S.: Research challenges for the security of control systems. In: Proceedings of USENIX Workshop on Hot Topics in Security (HotSec) (2008)
3. Caselli, M., Zambon, E., Kargl, F.: Sequence-aware intrusion detection in industrial control systems. In: Proceedings of the Workshop on Cyber-Physical System Security (CPSS), pp. 13–24. ACM (2015)
4. Chabukswar, R., Sinópoli, B., Karsai, G., Giani, A., Neema, H., Davis, A.: Simulation of network attacks on SCADA systems. In: Proceedings of Workshop on Secure Control Systems (2010)
5. Falliere, N., Murchu, L.O., Chien, E.: W32.Stuxnet dossier
6. Goldenberg, N., Wool, A.: Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *Int. J. Crit. Infrastruct. Prot.* **6**(2), 63–75 (2013)
7. John, K.H., Tiegelkamp, M.: IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids, 2nd edn. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-12015-2>
8. Karnouskos, S.: Stuxnet worm impact on industrial cyber-physical system security. In: Proceedings of Conference on Industrial Electronics Society (IECON), pp. 4490–4494. IEEE (2011)

9. Kim, D.-Y.: Cyber security issues imposed on nuclear power plants. *Ann. Nucl. Energy* **65**, 141–143 (2014)
10. Kim, G.H., Spafford, E.H.: The design and implementation of tripwire: a file system integrity checker. In: *Proceedings of the Conference on Computer and Communications Security (CCS)*, pp. 18–29. ACM (1994)
11. Kosut, O., Jia, L., Thomas, R., Tong, L.: Malicious data attacks on smart grid state estimation: attack strategies and countermeasures. In: *Proceedings of the IEEE Conference on Smart Grid Communications (SmartGridComm)*, pp. 220–225, October 2010
12. Krotofil, M., Cárdenas, A.A., Manning, B., Larsen, J.: CPS: driving cyber-physical systems to unsafe operating conditions by timing DoS attacks on sensor signals. In: *Proceedings of the Conference on Annual Computer Security Applications Conference (ACSAC)*, pp. 146–155. ACM (2014)
13. Lin, J., Yu, W., Yang, X., Xu, G., Zhao, W.: On false data injection attacks against distributed energy routing in smart grid. In: *Proceedings of Conference on Cyber-Physical Systems (ICCP)* (2012)
14. Liu, Y., Ning, P., Reiter, M.K.: False data injection attacks against state estimation in electric power grids. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **14**(1), 13 (2011)
15. McLaughlin, S.: On dynamic malware payloads aimed at programmable logic controllers. In: *Proceedings of USENIX Conference on Hot Topics in Security (HotSec)*, p. 10, August 2013
16. McLaughlin, S., McDaniel, P.: SABOT: specification-based payload generation for programmable logic controllers. In: *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 439–449. ACM (2012)
17. McLaughlin, S.E., Zonouz, S.A., Pohly, D.J., McDaniel, P.D.: A trusted safety verifier for process controller code. In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2014)
18. Milinkovic, S.A., Lazic, L.R.: Industrial PLC security issues. In: *Proceedings of Conference on Telecommunications Forum (TELFOR)*, pp. 1536–1539. IEEE (2012)
19. Morris, T.H., Gao, W.: Industrial control system cyber attacks. In: *Proceedings of the Symposium for ICS and SCADA Cyber Security Research (ICS-CSR)*. BCS Learning and Development Ltd. (2013)
20. Pollet, J.: Electricity for free? The dirty underbelly of SCADA and smart meters. In: *Proceedings of Black Hat USA* (2010)
21. Wang, E., Ye, Y., Xu, X., Yiu, S., Hui, L., Chow, K.: Security issues and challenges for cyber physical system. In: *Proceedings of Conference on Cyber, Physical and Social Computing (CPSCoM)*, pp. 733–738, December 2010
22. Zhu, B., Joseph, A., Sastry, S.: A taxonomy of cyber attacks on SCADA systems. In: *Proceedings of Conference on Cyber, Physical and Social Computing (CPSCoM)*, pp. 380–388 (2011)
23. Zonouz, S., Rogers, K., Berthier, R., Bobba, R., Sanders, W., Overbye, T.: SCPSE: security-oriented cyber-physical state estimation for power grid critical infrastructures. *IEEE Trans. Smart Grid* **3**(4), 1790–1799 (2012)
24. Zonouz, S., Rushi, J., McLaughlin, S.: Detecting industrial control malware using automated PLC code analytics. *IEEE Secur. Priv.* **12**(6), 40–47 (2014)

Enforcing Memory Safety in Cyber-Physical Systems

Eyasu Getahun Chekole^{1,2}(✉), John Henry Castellanos¹, Martín Ochoa¹,
and David K. Y. Yau^{1,2}

¹ Singapore University of Technology and Design, Singapore, Singapore
`eyasu_chekole@mymail.sutd.edu.sg`

² Advanced Digital Sciences Center, Singapore, Singapore

Abstract. Cyber-Physical Systems (CPS) integrate computations and communications with physical processes and are being widely adopted in various application areas. However, the increasing prevalence of cyber attacks targeting them poses a growing security concern. In particular, attacks exploiting memory-safety vulnerabilities constitute a major attack vector against CPS, because embedded systems often rely on unsafe but fast programming languages to meet their hard time constraints. A wide range of countermeasures has been developed to provide protection against these attacks. However, the most reliable countermeasures incur in high runtime overheads. In this work, we explore the applicability of strong countermeasures against memory-safety attacks in the context of realistic Industrial Control Systems (ICS). To this end, we design an experimental setup, based on a secure water treatment plant (SWaT) to empirically measure the memory safety overhead (MSO) caused by memory-safe compilation of the Programmable Logic Controller (PLC). We then quantify the tolerability of this overhead in terms of the expected real-time constraints of SWaT. Our results show high effectiveness of the security measure in detecting memory-safety violations and a MSO (197.86 μ s per scan-cycle) that is also tolerable for the SWaT simulation. We also discuss how different parameters impact the execution time of PLCs and the resulting absolute MSO.

1 Introduction

Cyber-physical systems [1–3], which integrate computations and communications with physical processes, are gaining attention and being widely adopted in various application areas including power grid, water systems, transportation, manufacturing, healthcare services and robotics, among others. Despite their importance, the increasing prevalence of cyber attacks targeting them poses a serious security risk. On the other hand, real-time requirements and legacy hardware/software limit the practicality of certain security solutions available. Thus,

The original version of this chapter was revised. Eyasu Getahun Chekole's name was corrected. An erratum to this chapter can be found at https://doi.org/10.1007/978-3-319-72817-9_18

the trade-off between security, performance and cost remains one of the main design challenges for CPS.

Cyber attacks in CPS can target either the communication network or the individual network nodes. Network attacks might sniff, drop, or compromise data packets as they traverse the communication channels, e.g., man-in-the-middle (MITM) and replay attacks. They exploit vulnerabilities in the communication paths or protocols. Attacks against computing nodes, e.g., PLCs, might exploit vulnerabilities in the firmware or control software. For example, malware can corrupt the memory of the PLC to hijack or otherwise subvert its operations. In this paper, we focus on the class of software/firmware attacks [4, 5] that compromises the integrity of the PLC's memory system to inject or trigger malicious code or data.

Memory safety vulnerabilities arise due to the use of programming languages where memory is handled manually such as C/C++. Those languages are popular and they are particularly relevant in systems with stringent real-time constraints since they allow skilled programmers to produce very efficient compiled code. However, programmers are also responsible to avoid potential security flaws in their code. Since firmwares of PLCs are commonly implemented in these languages, the memory unsafety represents a significant security concern. Real-world examples, such as buffer overflows and dereferences of dangling pointers, are regularly discovered and reported in modern PLCs.

For instance, Common Vulnerabilities and Exposures (CVE) [6] have been reported for a wide range of memory safety vulnerabilities on PLCs in the last couple of decades. For example, a buffer overflow vulnerability reported by CVE concerns Allen-Bradley's RSLogix Micro Starter Lite (CVE-2016-5814) [7], which allows remote attackers to execute arbitrary codes via a crafted rich site on summary (RSS) project file. Yet other buffer overflow vulnerabilities are reported on [8, 9] on this PLC. Similarly, a CVE has also recently reported memory safety vulnerabilities discovered on Siemens PLC [10, 11], Schneider Electric Modicon PLC [12, 13], ABB PLC automation [14], and so on.

Since such attacks regularly impact general IT systems as well, a wide range of countermeasures have been developed against memory-safety attacks. They differ by various characteristics including architecture, runtime overhead, type of memory errors covered, accuracy of detecting errors, platforms supported, etc. One can generally classify them into four categories: *probabilistic* (e.g., stack canaries, address space layout randomization (ASLR), position independent execution (PIE), [15, 16]), *paging-based* (e.g., non-executable memory page (NX)), *control-flow integrity (CFI) based* (e.g., [17–19]), and *code-instrumentation based* (e.g., ASan [20], SoftBoundCETS [21–26]).

Most of the countermeasures mentioned above are designed for desktop computers and servers with x86-based target architectures. When applied in a CPS context, they have the following limitations. First, almost all of them have architectural compatibility problems in working with RISC-based ARM or AVR CPU architectures. More fundamentally, many countermeasures have non-negligible runtime overheads, which might be prohibitive in the context of CPS. Hence, vis-a-vis the exploitation concerns, performance and availability are equally critical in a CPS environment.

Thus, to cover a wide range of memory safety violations, the code instrumentation based countermeasures, which we refer to as memory safety tools, offer stronger guarantees. These tools do not directly counter memory-safety attacks. Rather, they detect and mitigate memory safety violations before the attackers get a chance to exploit them. Numerous memory safety tools are available that differ by error coverage, accuracy, and performance. Although there are published benchmarks for the overheads caused by such tools, which give an intuition of average penalties to be paid when using them, it is still unclear how they perform in a CPS context.

In this research, after reviewing several available tools, we port a popular memory-safety compilation tool (ASan [20]) to work on a system architecture mimicking a realistic CPS. We adopt an empirical approach to measure its MSO, so that we can quantify its performance impact and hence acceptability for different applications. Our experiments are inspired by SWaT, a realistic CPS water system testbed that contains a set of real-world vendor-supplied PLCs. However, the vendor’s PLC firmware is closed-source; it does not allow us to incorporate additional memory safety solutions. Hence, we prototyped an experimental setup, which we call open-SWaT, based on open-source PLCs to mimic the behavior of the SWaT according to its detailed operational profile. We report experiments conducted on open-SWaT, which indicate that the MSO would not impact the normal operation of SWaT.

In summary, this work tackles the problem of *quantifying the practical tolerability of strong memory-safety enforcement on realistic Cyber-Physical Systems with hard real-time constraints and limited computational power*.

We make the following contributions: **(a)** We enforce a memory safety countermeasure based on secure compiling for a realistic CPS environment. **(b)** We empirically measure the tolerability of the induced overhead of the countermeasure based on the constraints of a real industrial control system. **(c)** We discuss parameters that affect the absolute overheads (in terms of time) in order to generalize our observations on tolerability beyond our case study.

2 Background

In this section, we provide background information on cyber-physical systems, the CPS testbed we use for experimenting (SWaT) and the memory safety tool we enforce to our CPS design (ASan).

2.1 Overview of CPS

Unlike traditional IT systems, CPS involve complex interactions among various entities while integrating physical plants and control devices (PLCs) via communication networks. These interactions are also constrained by hard deadlines. Missing deadlines could result in disruption of control loop stability or a complete system failure, in the worst case. This makes CPS highly real-time constrained systems. CPS are also highly resource-constrained systems. Edge devices,

e.g., PLCs and input/output devices, have limited memory size and CPU speed. In general, as shown on Fig. 1, CPS consists of the following entities:

- *Plants*: entities where physical processes take place.
- *Sensors*: devices that observe or measure state information of plants and physical processes, which will be used as inputs for PLCs.
- *PLCs*: entities that make decisions and issue control commands (based on inputs obtained from sensors) to control plants.
- *Actuators*: entities that implement the control commands issued by PLCs.
- *Communication networks*: communication medias where packets (containing sensor measurements, control commands, alarms, diagnostic information, etc.) transmit over from one entity to another.
- *SCADA*: an entity designed for process controlling and monitoring. It consists of human-machine interface (HMI) – for displaying state information of plants and historian server – for storing all operating data, alarm history and events.

2.2 Overview of SWaT

SWaT is a fully operational water purification plant designed for research in the design of secure cyber-physical systems. It produces 5 gal/min of doubly filtered water.

Purification Process. The whole water purification process is carried out by six distinct, but cooperative, sub-processes. Each process is controlled by an independent PLC (details can be found on [27]).

Components and Specifications. The design of SWaT consists of the following components and system specifications:

- *PLCs*: six redundancy closed-source Allen Bradley ControlLogix L5571 (1756-L71) PLCs are deployed to control each sub-process. They communicate one another via EtherNet/IP-CIP (Common Industrial Protocol).
- *Real-time constraint*: the real-time constraint of SWaT is 10 ms. The notion of real-time constraint (in the context of CPS) is discussed in detail on Sect. 3.5.
- *Remote input/output (RIO)*: remote terminals consisting of digital inputs (DI), analog inputs (AI) and digital outputs (DO). RIO of SWaT contains 32 DI (water level and pressure switches), 13 AI (water pressure, flow rate and water level sensors), and 16 DO (actuators such as pumps and motorized valves).

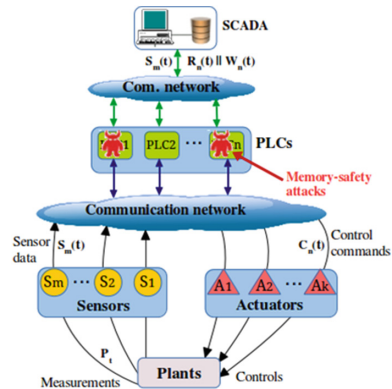


Fig. 1. The CPS architecture and memory-safety attacks

- *Communication frequency*: the six PLCs exchange packets among each other and with the SCADA system depending on operational conditions. If we take the busiest PLC (PLC2), it transfers as high rate as 382 packets per second with its most active peer (PLC3) or as low as three packets per second to another. Regarding connections with all devices in SWaT, we estimate that the busiest PLC (PLC2) sends and receives requests to a ratio of 1000 packets per second.
- *PLC program*: a complex control logic written in ladder language generally. It consists of various instructions (see the list on Table 1). PLC2 (the busiest PLC) has a complex PLC program consisting of 127 instructions.
- *SCADA system*: a touch panel, an HMI, is mounted to SWaT to provide users a local system supervisory monitoring and controls.

2.3 ASan

As discussed in the introduction, despite several memory safety tools being available, their applicability in the CPS environment is limited due to compatibility and performance reasons. After researching and experimenting on various memory safety tools, we chose ASan [20] as a basis for our empirical study because of its error coverage, high detection accuracy and relatively low runtime overhead.

ASan is a compile-time code instrumentation memory safety tool. It inserts memory safety checks into the program code at compile-time, and it detects and mitigates memory safety violations at runtime. ASan covers several memory safety vulnerabilities such as buffer overflows, dangling pointers (use after free), use after return, memory leaks and initialization order bugs. Although there are also some memory errors, e.g., uninitialized memory reads, that are not covered by ASan, such errors are less critical and rarely exploited in practice.

ASan has relatively low overhead when compared to other code instrumentation tools. Different level of performance optimization options are also available for the tool. A detailed account on error coverage and runtime overhead of ASan (in comparison with other tools) is provided on [20].

Similar to other memory safety tools, the off-the-shelf ASan has compatibility issues with RISC-based ARM or AVR based architectures. ASan has also a problem of dynamically linking shared libraries, e.g., `glibc`, for our experimental setup. Therefore, as explained on Sect. 4.2, our initial task was fixing those problems to fit our experimental design. For this task it was crucial that ASan is an open-source project, which allowed for several customizations.

3 System and Attacker Model

We have seen high-level overview of CPS and its architecture on Sect. 2.1. In this section, we model the interactions of various entities in the CPS, attack scenarios targeting them, and MSO and its tolerability for a given CPS.

3.1 Modeling Interactions

Entities of CPS (as shown on Fig. 1) interact one another in a continuous and loop-back manner. Suppose the state vector of plant P at time t is $P_t \in \mathbb{R}^k$, where k is dimensions of state variables of the plant such as water level, water flow, water pressure, etc. Sensor S_m measures the state information $S_m(t) \in \mathbb{R}$ at time t and sends to PLCs. When PLC_n receives the state information $S_m(t)$ from sensor S_m , it makes decision and issues control command $C_n(t) \in \mathbb{R}^q$, where q is dimensions of control variables for different actuators such as pumps, valves, etc., in accordance with its control logic. Thus, $C_n(t) = f(S_m(t))$, where f is the control logic of PLC_n . Then, it sends this control command to actuators. Actuators will take actions to implement the command. The actions taken by the actuators will change the state information of the plant. The loop-back routine will continue in a similar way.

The SCADA system directly communicates with PLCs via a specific communication protocol, e.g., ENIP, Modbus, and CIP. It can send read command ($R_n(t) \in \mathbb{R}$) or write command ($W_n(t) \in \mathbb{R}$) to PLC_n at time t . With $R_n(t)$, it requests state information of the plant. PLCs will then feed it with the information received from sensors. The SCADA system will then display the state information to users via its integrated HMI. With $W_n(t)$, (users via) the SCADA system can also issue control commands to change state information of the plant. Upon receiving, PLCs will request actuators to implement it. The whole CPS process is a continuous process triggered and synchronized by system time.

3.2 Attacker Model

As discussed in the introduction, memory-safety attacks such as code injection and code reuse mainly exploit memory safety vulnerabilities on firmwares of the PLCs such as buffer overflows, dangling pointers, and so on. Their main goal is to take control of PLC's programs, and eventually affect the CPS. Bittau et al. [28] provided detail information on how to exploit memory safety vulnerabilities, develop an attack technique and then take control of the vulnerable system. In general, there are five main steps involved in the memory-safety attacks scenario.

1. Interacting with the PLC, e.g., via network connection (for remote attacks).
2. Finding a memory safety vulnerability in the firmware/control software so that will be exploited accordingly, e.g., buffer overflow vulnerability.
3. Triggering a memory safety violation to happen on the PLC at runtime, e.g., overflow the buffer.
4. Overwriting important addresses of the vulnerable program, e.g., overwrite return address of the PLC program.
5. Using the new return address, diverting control flow of the program to an injected (malicious) code (in case of code injection attacks) or to existing modules of the vulnerable program (in case of code reuse attacks). In the former case, the attacker can get control of the PLC with its injected code. In the later case, the attacker still needs to collect important gadgets from the program (basically by scanning the program's text segment), then he will synthesis a shellcode that will allow him to get control of the PLC.

After getting control of the PLC, the attacker would be able to make changes on normal operations of the PLC. For example, by changing the control logic he can force the PLC to issue tainted control command $C'_n(t)$.

$$C'_n(t) = f'(S_m(t)), f \neq f' \quad (1)$$

where f' is the tainted control logic by the attacker. Actuators will be then forced to implement the control command issued by the attacker. This will change the operational behavior of the PLC, and so does the CPS, in general. Figure 1 shows an architectural point of view of memory-safety attacks in CPS.

3.3 Modeling System Performance

Overall performance of CPS can be affected by communication latencies such as latencies of packets carrying sensor measurements ($L_{S_m(t)} \in \mathbb{R}$) and latencies of packets carrying control commands ($L_{C_n(t)} \in \mathbb{R}$). In addition, it can be also affected by the execution time that will be taken by each entity to complete its respective tasks. Sensors take some execution time ($T_{sensors} \in \mathbb{R}$) to measure or observe state information of plants. PLCs also take substantial time ($T_{PLCs} \in \mathbb{R}$) to make decisions and issue control commands that involves complex cyclic operations (details on Sect. 3.4). Similarly, actuators take some execution time ($T_{actuators} \in \mathbb{R}$) to implement control commands. Therefore, the overall CPS time ($T_{CPS} \in \mathbb{R}$) can be intuitively represented with the following formula:

$$T_{CPS} = L_{S_m(t)} + L_{C_n(t)} + T_{sensors} + T_{PLCs} + T_{actuators} \quad (2)$$

However, accurately modeling the overall performance of the dynamics of CPS is not that much intuitive since the interactions among the entities are very complex and some of them are non-deterministic. As described on Sect. 5.4, it can be also affected by various internal and external factors. Zhang et al. [29] modeled an approximation of the dynamics of CPS as a general difference equations with state variables, control variables, and the noises. However, we do not need to go in detail on this since the main focus of this paper is modeling memory safety overheads rather than overall performance of CPS. Therefore, we will mainly focus on T_{PLCs} on the following section since it is directly involved in the MSO computation.

3.4 Modeling MSO

To ensure memory safety, firmwares of PLCs should be compiled with a memory safety tool. Hence the memory safety overhead will be added to the execution time of PLCs, i.e., T_{PLCs} . To model T_{PLCs} , we need to clearly understand how PLCs operate in CPS. PLCs handle two main processes – a communication process and a scan cycle process (the terms are proposed by the authors in the context of the conducted research). These processes are handled by two separate and parallel threads – a communication thread and a scan cycle thread,

respectively. The communication thread handles any network communication related tasks, e.g., creating connections with communicating entities, receiving and sending network requests, etc. It is a continuous process. On the other hand, the scan cycle thread handles the main PLC process that involves three operations: scan inputs, execute control program and update outputs. The PLC scan cycle starts by reading state of all inputs from sensors and storing them to the PLC input buffer. Then, it will execute the PLC's control logic and issue control commands according to the state of sensor inputs. The scan cycle will be then concluded by updating output values to the output buffer and sending control commands to the actuators. Unlike the communication process, the PLC scan process is a cyclic process. Before modeling MSO, it is important to describe the following notions that involve in the overall control process:

- *Cycle time* ($T_c \in \mathbb{R}$): an upper bound time set for a scan cycle, i.e., a scan cycle must be completed within this time period. When the cycle time is over, the next scan cycle will start immediately.
- *Scan time* ($T_s \in \mathbb{R}$): the measurement of the actual time elapsed by the PLC scan cycle process.

$$T_s = T_{si} + T_{ep} + T_{uo}, \quad (3)$$

where $T_{si} \in \mathbb{R}$ is time elapsed to scan inputs, $T_{ep} \in \mathbb{R}$ is time elapsed to execute PLC program and $T_{uo} \in \mathbb{R}$ is time elapsed to update outputs.

- *Buffer waiting time* ($T_{bw} \in \mathbb{R}$): the communication and scan cycle threads can run in parallel, but they access shared resources, i.e., input and output buffers. When the two threads try to access those shared buffers, race condition or deadlock could happen. To avoid that happening, a mutual exclusion buffer locking function is introduced to PLCs that prevents any attempt of simultaneous access to the shared buffers. This will introduce some waiting time. Suppose (T_{bw}) is the waiting time of the scan cycle thread. Computing T_{bw} is non-deterministic; because the locking process is not deterministic since it involves a stochastic communication process. This buffer waiting time will directly affect the PLC scan time. Therefore, the overall PLC scan time (T_s) can be rewritten as follows,

$$T_s = T_{si} + T_{ep} + T_{uo} + T_{bw} \quad (4)$$

- *Thread sleeping time* ($T_{ts} \in \mathbb{R}$): if the scan cycle is completed before the allocated cycle time, the thread will sleep for the remaining cycle time.

By design, $T_s + T_{ts} \leq T_c$ unless otherwise execution time of the PLC is affected by external factors, e.g., MSO. The memory safety overhead ($MSO \in \mathbb{R}$) is the average difference in scan time for a PLC firmware that is compiled with and without a memory safety tool. As we will see in the following, using average time is justified by the small standard deviation in multiple measurements.

$$MSO = \hat{T}_s - T_s, \quad (5)$$

where \hat{T}_s is the scan time when the firmware is compiled with memory-safety.

Computing MSO is also non-deterministic since the scan time (T_s or \hat{T}_s) computation is non-deterministic because of T_{bw} . Obviously, non-deterministic computations are less precise as compare to actual measurements. Thus, given the fact that the real-time constraints of CPS are in the order of milliseconds (often 3–10 ms), there would no better approach than the empirical one to precisely measure MSO and quantify its tolerability. That is the main reason why we proposed the empirical approach for this research.

3.5 Quantifying Tolerability

We define MSO tolerability with respect to real-time constraints of CPS. Real-time constraints of CPS often defined based on the real-time constraints of its respective PLCs. Real-time constraint of PLCs is defined with its respective cycle time. As discussed on Sect. 3.4, an upper bound time, i.e., T_c , will be set for each scan cycle. Meaning, each scan cycle of the PLC must be completed within the cycle time duration specified for that PLC, i.e., $T_s \leq T_c$. This constraint is called *real-time constraint* of the PLC.

But, as we discussed above, the memory safety overhead increases the PLC scan time. If the scan time with memory safety enabled, i.e., \hat{T}_s , still respects real-time constraint of the PLC, then the memory safety overhead can be considered as *tolerable*. Therefore, we define tolerability of MSO in average-case scenario ($T(MSO)$), i.e., when tolerability is quantified based on an average \hat{T}_s of n scan cycle measurements, as follows,

$$T(MSO) = \begin{cases} \text{Tolerable,} & \text{if } \text{mean}(\hat{T}_s) \leq T_c \\ \text{Not tolerable,} & \text{otherwise} \end{cases} \quad (6)$$

where $\text{mean}(\hat{T}_s) = \frac{\sum_{i=1}^n \hat{T}_s(i)}{n}$, where $\hat{T}_s(i)$ denotes measurement of the i -th scan cycle performed with a memory-safe compilation and n is the total number of scan cycles performed.

We often use the average-case scenario to quantify tolerability. However, a single scan time could potentially violate timing constraint of the PLC. Therefore, it is also important to consider the worst-case scenario, i.e., with the maximum \hat{T}_s of n measurements, to validate if the MSO is fully tolerable. Thus, tolerability of the MSO in the worst-case scenario ($T(WMSO)$) can be defined as,

$$T(WMSO) = \begin{cases} \text{Tolerable,} & \text{if } \text{max}(\hat{T}_s) \leq T_c \\ \text{Not tolerable,} & \text{otherwise} \end{cases} \quad (7)$$

where $\text{max}(\hat{T}_s) = \text{max}(\{\hat{T}_s(1), \dots, \hat{T}_s(n)\})$.

The choice of scenarios may depend on delay-sensitivity of the CPS or users' preference. For SWaT, we use the average-case as a basis to validate tolerability. But, we also use the worst-case to validate if it is fully tolerable. If it is not, then we do further investigations to figure out if the root cause is actually the MSO or other exceptions, e.g., sudden service interruptions due to unforeseen reasons.

4 Experimental Design

As discussed in the introduction, SWaT is based on closed-source PLCs. Thus, we designed open-SWaT – a CPS based on open source PLCs that mimics features and behaviors of SWaT. We discuss design details of open-SWaT and our experimental results in the following.

4.1 open-SWaT

open-SWaT is designed using OpenPLC [30] – an open source PLC for industrial control systems. With open-SWaT, we reproduce operational details of SWaT; in particular we reproduce the main characteristics (mentioned on Sect. 5.4) that have a significant impact on the scan time and MSO:

PLCs: we designed the PLCs using an OpenPLC controller that runs on top of Linux on Raspberry PI. To reproduce hardware specifications of SWaT PLCs, we specified 200 MHz fixed CPU speed and 2 MB user memory for our PLCs.

RIO: we use Arduino Mega as RIO terminal. It has AVR based processor with 16 MHz clock speed. To reproduce the number of I/O devices of SWaT, we used 32 DI (push-buttons, switches and scripts), 13 AI (temperature and ultrasonic sensors) and 16 DO (light emitter diodes (LEDs)).

PLC program: we designed a control logic in ladder diagram that has similar complexity to the one in SWaT (a sample diagram is shown on Fig. 3). It consists of various types of instructions such as logical (AND, OR, NOT, SR (set-reset latch)), arithmetic (addition (ADD), multiplication (MUL)), comparisons (equal (EQ), greater than (GT), less than (LT), less than or equal (LE)), counters (up-counter (CTU)), timers (turn on timer (TON), turn off timer (TOF)), contacts and coils (normally-open (NO), normally-closed (NC)). The overall PLC program consists of 129 instructions (see details on Table 1).

Communication frequency: A high-level architecture of open-SWaT is shown on Fig. 2. open-SWaT uses both type of modbus communication protocols – modbus TCP (for Ethernet or wireless communication) and modbus RTU (for serial communication). Frequency of communication among PLCs and the SCADA system is similar to that in SWaT. The communication between PLCs and Arduino is via USB serial communication. The frequency of receiving inputs and sending outputs with Arduino is 100 Hz.

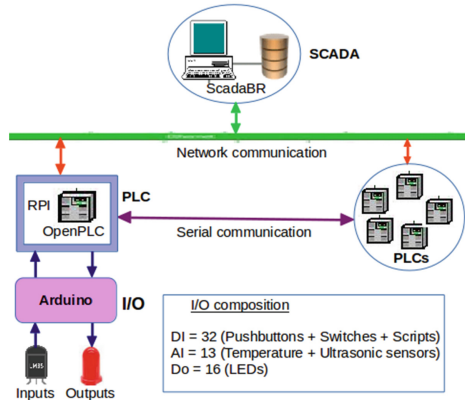


Fig. 2. Architecture of open-SWaT

Table 1. Instruction count

Instructions	Count
Logical	
AND	17
OR	14
NOT	5
SR	1
Arithmetic	
ADD	1
MUL	2
Comparisons	
EQ	3
GT	3
LT	2
LE	2
Timers	
TON	3
TOF	9
Counters	
CTU	1
Selections	
SEL	1
MAX	1
Contacts	
NO	38
NC	3
Coils	
NO	21
NC	2
Total	129

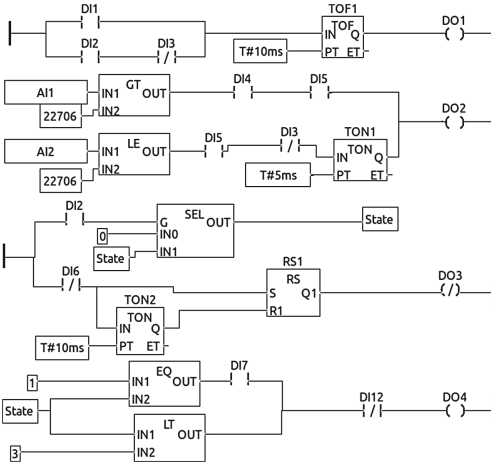


Fig. 3. Sample PLC program in ladder diagram

Real-time constraints: based on the real-time constraint of SWaT, we set a 10 ms cycle time (real-time constraint) to each PLC in open-SWaT.

SCADA system: we use ScadaBR [31], a full SCADA system consisting of a web-based HMI.

In summary, the design of open-SWaT is, by design, very close to SWaT. In particular, the inputs PLCs receive from sensors, the logic they execute, the number of nodes they are communicating with, the frequency of communications, etc., are designed to be similar. Thus, we can expect that the MSO in open-SWaT would also remain close to that in SWaT. Therefore, if the MSO is tolerable in open-SWaT, it would be the same for SWaT. In the future, as a further validation step, we may replace some PLCs at SWaT with the open-source and memory-safety enabled PLCs of open-SWaT to evaluate its functional equivalence.

4.2 Memory Safety Compilation

As stated on the introduction, our approach to counter memory-safety attacks is by secure compiling of the PLCs’ firmware with a compile-time code-instrumentation tool. We ported ASan for that, but porting ASan to our CPS design was not a straightforward task because of its compatibility and dynamic library linking problems. Thus, we fixed those problems by modifying and rebuilding its source code and by enabling dynamic library linking runtime options.

To do secure compilation, we also need to integrate ASan with a native C/C++ compiler. Fortunately, ASan can work with GCC or CLANG with a `-fsanitize=address` switch – a compiler flag that enables ASan at compile time. Various types of compile-time and runtime flags (e.g., performance optimization, stack trace, diagnostics, specific error checking and blacklist file) are also available. Therefore, we compiled our PLC using GCC with ASan enabled.

4.3 Detection and Mitigation

As discussed on Sect. 2.3, ASan instruments the protected program to ensure that memory access instructions never read or write the so called “poisoned” redzones [20]. Redzones are small regions of memory inserted in between any two stack, heap or global objects. Since the program should never address them, access to them indicates an illegal behavior and it will be considered as memory-safety violation. This policy detects sequential buffer over/underflows, and some of the more sophisticated pointer corruption bugs such as dangling pointers (use after free) and use after return bugs (see the full list on Table 3). With this enforcement, we detected two global buffer overflow vulnerabilities on the OpenPLC Modbus implementation.

ASan’s mitigation approach is based on the principle of “automatically aborting” the vulnerable program whenever a memory-safety violation is detected. It is effective in restricting memory-safety attacks not to be able to exploit the vulnerabilities. However, this might not be acceptable in a CPS environment since the abortion affects system availability, and potentially controllability of a process. Thus, we are currently working on advanced mitigation strategies to address these limitations.

4.4 Experimental Results

We have implemented a function using POSIX clocks (in nanosecond resolution) that measures execution time of each operation in the PLC scan cycle. Table 2 summarizes overall performance of the PLC including execution time of each PLC operation and its respective memory safety overheads. For statistical analysis, we have also included minimum (min), maximum (max), mean, variance and standard deviation (sd) of each operation.

Table 2. Memory safety overhead (MSO)

Operations	Number of scan cycles	Network devices	T_s (in μs)						\hat{T}_s (in μs)		Average MSO	
			mean	min	max	mean	var	sd	in μs	in %		
Input scan	10×1000	6	50.657	69.740	1997.348	112.380	8587.425	92.668	61.723	121.85		
Program execution	10×1000	6	79.537	104.323	1325.732	176.723	7799.361	88.314	91.19	122.19		
Output update	10×1000	6	111.635	107.552	1518.909	150.583	5503.488	74.185	38.95	34.89		
Full scan time	10×1000	6	241.829	281.615	4841.989	439.686	7296.758	85.056	197.86	81.82		

5 Evaluation and Discussion

After conducting experiments, we performed a detailed evaluation to figure out whether the memory safety tool is accurate enough to detect memory safety violations and efficient enough to work in a CPS environment. In brief, our evaluation has three parts: *security (accuracy)* – detection accuracy of ASan, *performance (efficiency)* – tolerability of its runtime overhead in CPS, and *memory usage* overheads.

5.1 Security

As a sanity check on our configuration, we have evaluated our setup against the memory safety vulnerabilities listed on Table 3, to explore detection accuracy of ASan in the CPS environment. The results show, as in the original paper [20], the tool detects memory safety violations with high accuracy – with no false positives for all vulnerabilities, and very few false negatives for global buffer overflows and use after return vulnerabilities. The paper [20] briefly discussed the conditions how and where the false negatives could happen. According to [20], ASan performs better than the other memory safety tools in terms of error coverage and accuracy of detecting them. ASan also effectively mitigates the detected violations regardless of the limitations discussed on Sect. 4.3.

Table 3. Detection accuracy

Vulnerabilities	False positive	False negative
Stack buffer overflow	No	No
Heap buffer overflow	No	No
Global buffer overflow	No	Rare
Dangling pointers	No	Rare
Use after return	No	No
Initialization order bugs	No	No
Memory leaks	No	No

5.2 Performance

According to published benchmarks [20], the average runtime overhead of ASan is about 73%. However, all measurements were taken on a non-CPS environment.

In our setting, the average overhead is $197.86 \mu\text{s}$ (81.82%) as shown on Table 2. To validate its tolerability for our system, i.e., SWaT, we have checked if it satisfies the tolerability conditions defined on Eq. (6) (for average-case) and Eq. 7 (for worst-case). As demonstrated on Table 2, $mean(\hat{T}_s) = 439.686 \mu\text{s}$, and $T_c = 10000 \mu\text{s}$. Therefore, according to Eq. (6), the MSO is tolerable for SWaT with the average-case scenario.

To validate the tolerability in the worst-case scenario, we check if it satisfies Eq. 7. As shown on Table 2, $max(\hat{T}_s) = 4841.989 \mu\text{s}$, and $T_c = 10000 \mu\text{s}$. It is still tolerable, thus ASan satisfies the real-time constraint of SWaT both in the average-case and worst-case scenarios. Therefore, we conclude that SWaT would fully tolerate the overhead caused by memory-safe compilation, while significantly increasing its security.

5.3 Memory Usage

We also evaluated memory usage overheads of our security measure. Table 4 summarizes the increase in memory usage, stack size, binary size and shared library usage collected by reading VmPeak, VmStk, VmExe and VmLib fields, respectively, from `/proc/self/status`. There is significant increase in

memory usage due to the allocation of large redzones with `malloc`. This overhead is still acceptable since most PLCs come with at least 1 GB memory size.

Table 4. Memory overheads (in MB)

Category	Original	Instrumented	Increase
Memory usage	22.516	852.480	$37.86\times$
Stack size	0.136	0.142	$1.04\times$
Binary size	0.140	0.328	$2.34\times$
Shared library usage	2.796	3.372	$1.21\times$

5.4 Sensitivity Analysis

There are many factors, that can affect the scan time or MSO of a PLC. But, as we verified it experimentally, the main variables that significantly affect the scan time and MSO are (i) the number of input devices (sensors) connected to the PLC ($N_S \in \mathbb{N}$) (ii) the number of output devices (actuators) connected to the PLC ($N_A \in \mathbb{N}$) (iii) complexity of the PLC program ($C_P \in \mathbb{R}^z$, where $z = \{\text{number of instructions, type of instructions}\}$) (iv) frequency of communications (frequency of packets) between the PLC and other entities ($F_P \in \mathbb{R}$) and (v) CPU speed of the PLC ($S_{CPU} \in \mathbb{R}$). F_P is dependent on the number of connections the PLC is communicating with. We have done a preliminary sensitivity analysis on these variables regarding its effect on the PLC scan time and MSO. The variables affect execution time of the PLC operations discussed on Sect. 3.4. N_S affects input scan time, i.e., T_{si} , of the PLC. N_A affects output update time, i.e., T_{uo} , of the PLC. As discussed on Sect. 4.1, the PLC program consists of various type of instructions. Each instruction has different execution time. Therefore, C_P can be determined by the number and the type of instructions it consists of; and it affects the program execution time, i.e., T_{ep} . F_P affects the buffer waiting time, i.e., T_{bw} , hence it affects T_{si} , T_{ep} and T_{uo} (or T_s in general). S_{CPU} affects all operations of the PLC.

Table 5. Sensitivity analysis

Logic complexity	Num. of scan cycles	CPU speed (in MHz)	Num. of sensors	Number of actuators	Num. of connections	Average T_s (in μs)	Average T_s^2 (in μs)	Average MSO	
								in μs	in %
Simple program (containing 4 instructions)	10×1000	200	2	2	0	190.74	292.26	101.52	53.22
					2	208.67	318.33	109.66	52.55
					4	203.72	330.41	126.69	62.19
					6	213.65	351.47	137.82	64.51
Complex program (containing 129 instructions)	10×1000	200	45	16	0	222.83	339.55	116.72	52.38
					2	232.15	376.12	143.97	62.02
					4	236.84	420.47	183.63	77.53
					6	241.83	439.69	197.86	81.82

Because of space limitation, we present here only the preliminary experiment we have done by involving two kinds of PLC logic: a simple PLC program containing 4 instructions (interacting with 2 sensors and 2 actuators) and a more complex program consisting of 129 instructions (interacting with 45 digital and analog sensors and 16 actuators), for different networking configurations (0 to 6 connections) to explore the impact of each variable in the scan time and the MSO computation. Our experimental result is shown on Table 5.

6 Related Work

In this section, we explore related works done in providing memory safety solutions against memory-safety attacks and measuring and analyzing memory safety over heads in the CPS environment.

As discussed in the introduction, many memory safety tools are available designed for general IT systems. However, they have limitations to work in the CPS environment. SoftBoundCETS is a compile-time code-instrumentation tool that detects wide range of spatial memory safety (SoftBound [21]) and temporal memory safety (CETS [22]) violations in C. However, its runtime overhead is very high (116%) as compare to ASan (73%). In addition, it is incompatible for the CPS environment; because it is implemented only for the x86-64 target architecture and it is also dependent on the LLVM infrastructure.

Coopriider et al. [32] enforced efficient memory safety solution for TinyOS applications by integrating Deputy [33], an annotation based type and memory safety compiler, with nesC [34], a C compiler. Thus, they managed to detect memory safety violations with high accuracy. To make this memory safety solution practical in terms of CPU and memory usage, they did aggressive optimization by implementing a static analyzer and optimizer tool, called cXprop. With cXprop, they managed to reduce memory safety overhead of Deputy from 24% to 5.2%, and they also improved memory usage through dead code elimination. However, their solution has limitations to apply it in a CPS environment, because it is dependent on runtime libraries of TinyOS.

Zhang et al. [29] modeled the trade-off between privacy and performance in CPS. While they leveraged the differential privacy approach to preserve

privacy of CPS, they also analyzed and modeled its performance overhead. They proposed an approach that optimizes the system performance while preserving privacy of CPS. This work is interesting from point of view of analyzing performance overheads in CPS, but not from the memory safety perspective.

Stefanov et al. [35] proposed a new model and platform for the SCADA system of an integrated CPS. With the proposed platform, they modeled real-time supervision of CPS, performance of CPS based on communication latencies, and also he assessed communication and cyber security of the SCADA system. He followed a generic approach to assess and control various aspects of the CPS. However, he did not specifically work on memory-safety attacks or MSO.

Several CFI based solutions (e.g., [18,19]) have been also developed against memory safety attacks. However, CFI based solutions have some limitations in general (i) determining the required control flow graph (often using static analysis) is hard and requires a significant amount of memory; (ii) attacks that do not divert control flow of the program cannot be detected (for instance using Data Oriented attacks [36]). These and other reasons can limit the applicability of CFI solutions in the CPS environment.

In summary, to the best of our knowledge, there is no prior research work done that enforced memory safety solutions specifically to the CPS environment, and that measured and analyzed tolerability of memory safety overheads in accordance to real-time constraints of cyber-physical systems.

7 Conclusion

In this work we presented the results of implementing a strong memory safety enforcement in a simulated albeit realistic industrial control system using ASan. Our setup allowed us to benchmark and empirically measure the runtime overhead of the enforcement and, based on the real-time constraints of an ICS, to judge the applicability in a realistic scenario. Our experiments show that the real-time constraint of SWaT can be largely met even when implementing strong memory safety countermeasures in realistic hardware. We also preliminary discuss what factors impact the performance of such a system, in a first attempt to generalize our results.

In the future, we intend to study other CPS with different constraints, e.g., in power grid systems, water distribution or smart home devices. Such studies will allow us to extrapolate formulas predicting the tolerability of systems to MSO and thus aiding in the design of resilient CPS before such systems are deployed.

References

1. Sha, L., Gopalakrishnan, S., Liu, X., Wang, Q.: Cyber-physical systems: a new frontier. In: SUTC 2008 (2008)
2. Lee, E.A., Seshia, S.A.: Introduction to Embedded Systems - A Cyber-Physical Systems Approach, 2nd edn. Version 2.0 edn. (2015)
3. Lee, E.A.: Cyber physical systems: design challenges. In: ISORC 2008 (2008)

4. Basnight, Z., Butts, J., Lopez, J., Dube, T.: Firmware modification attacks on programmable logic controllers. *IJCIP* **6**(2), 76–88 (2013)
5. Cui, A., Costello, M., Stolfo, S.J.: When firmware modifications attack: a case study of embedded exploitation. In: *NDSS 2013* (2013)
6. MITRE: Common Vulnerabilities and Exposures. <https://cve.mitre.org/>
7. CVE-5814. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5814>
8. CVE-6438. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6438>
9. CVE-6436. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6436>
10. CVE-0674. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-0674>
11. CVE-1449. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-1449>
12. CVE-0929. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0929>
13. CVE-7937. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-7937>
14. CVE-5007. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-5007>
15. Berger, E.D., Zorn, B.G.: Diehard: probabilistic memory safety for unsafe languages. In: *PLDI 2006* (2006)
16. Novark, G., Berger, E.D.: Dieharder: securing the heap. In: *CCS 2010* (2010)
17. Abadi, M., Budiu, M., Erlingsson, U., Ligatti, J.: Control-flow integrity. In: *CCS 2005*, pp. 340–353 (2005)
18. Zhang, M., Sekar, R.: Control flow integrity for cots binaries. In: *USENIX 2013* (2013)
19. Tice, C., Roeder, T., Collingbourne, P., Checkoway, S., Erlingsson, Ú., Lozano, L., Pike, G.: Enforcing forward-edge control-flow integrity in GCC & LLVM. In: *USENIX 2014*, pp. 941–955 (2014)
20. Serebryany, K., Bruening, D., Potapenko, A., Vyukov, D.: Addresssanitizer: a fast address sanity checker. In: *USENIX ATC 2012* (2012)
21. Nagarakatte, S., Zhao, J., Martin, M.M., Zdancewic, S.: SoftBound: highly compatible and complete spatial memory safety for C. In: *PLDI 2009* (2009)
22. Nagarakatte, S., Zhao, J., Martin, M.M., Zdancewic, S.: CETS: compiler enforced temporal safety for C. In: *ISMM 2010* (2010)
23. Simpson, M.S., Barua, R.K.: MemSafe: ensuring the spatial and temporal memory safety of C at runtime. *Softw. Pract. Experience* **43**(1), 93–128 (2013)
24. Bruening, D., Zhao, Q.: Practical memory checking with Dr. Memory. In: *CGO 2011* (2011)
25. Necula, G.C., Condit, J., Harren, M., McPeak, S., Weimer, W.: CCured: type-safe retrofitting of legacy software. *ACM Trans. Program. Lang. Syst.* **27**(3), 477–526 (2005)
26. Eigler, F.Ch.: Mudflap: pointer use checking for C/C++. Red Hat Inc. (2003)
27. Ahmed, C.M., Adepu, S., Mathur, A.: Limitations of state estimation based cyber attack detection schemes in industrial control systems. In: *SCSP-W 2016* (2016)
28. Bittau, A., Belay, A., Mashtizadeh, A., Mazières, D., Boneh, D.: Hacking blind. In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy* (2014)
29. Zhang, H., Shu, Y., Cheng, P., Chen, J.: Privacy and performance trade-off in cyber-physical systems. *IEEE Netw.* **30**(2), 62–66 (2016)
30. OpenPLC. <http://www.openplcproject.com/>
31. ScadaBR. <http://www.scadabr.com.br/>
32. Coopriider, N., Archer, W., Eide, E., Gay, D., Regehr, J.: Efficient memory safety for TinyOS. In: *SenSys 2007*, pp. 205–218 (2007)
33. The Deputy project (2007). <http://deputy.cs.berkeley.edu>
34. Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., Culler, D.: The nesC language: a holistic approach to networked embedded systems. In: *PLDI 2003* (2003)

35. Stefanov, A., Liu, C.C., Govindarasu, M., Wu, S.S.: Scada modeling for performance and vulnerability assessment of integrated cyber-physical systems. *Int. Trans. Electr. Energy Syst.* **25**(3), 498–519 (2015)
36. Hu, H., Shinde, S., Adrian, S., Chua, Z.L., Saxena, P., Liang, Z.: Data-oriented programming: on the expressiveness of non-control data attacks. In: *SP 2016* (2016)

Detecting Attacks in Industrial Control and Cyber-Physical Systems

Supporting the Human in Cyber Defence

Kirsi Helkala¹(✉), Benjamin J. Knox¹, Øyvind Jøsok^{1,2}, Ricardo G. Lugo³, Stefan Sütterlin^{4,5}, Geir Olav Dyrkolbotn^{1,6}, and Nils Kalstad Svendsen⁷

¹ Norwegian Defence Cyber Academy, Lhmr, Norway
khelkala@mil.no, {bknox,ojosok}@cyfor.mil.no

² Child and Youth Participation and Development Research Program,
Inland Norway University of Applied Sciences, Lhmr, Norway

³ Department of Psychology, Inland Norway University of Applied Sciences,
Lhmr, Norway

ricardo.lugo@inn.no

⁴ Faculty for Health and Welfare Sciences, Østfold University College,
Halden, Norway

stefan.sutterlin@hiiof.no

⁵ Center for Clinical Neuroscience, Oslo University Hospital, Oslo, Norway

⁶ Centre for Cyber and Information Security, Gjøvik, Norway

geir.dyrkolbotn@ccis.no

⁷ Norwegian University of Science and Technology, Gjøvik, Norway

nils.svendsen@ntnu.no

Abstract. Incident detection is not merely the result of a technological process, but the output of a socio-technical system where the human has an important part to play. In this paper we focus on the human role within a socio-technically defined incident detection context by discussing the case of the Norwegian Cyber Defence approach. We show that the human has an important part in the process, not only by owning technical skills but also high-level cognitive skills that help critical thinking, decision-making and communication. We further summarize the results of our previous research and discuss how it can be applied, in order to improve educational content of an incident detection team. We strongly believe that the topics discussed in this paper, when implemented and applied, will help transforming the weakest link - the human - to the strongest defence.

Keywords: Cyber security · Cyber defence · Human factors
Incident detection · Socio-technical system

1 Introduction

In this paper we focus on the human side of incident detection by reviewing the Computer Network Defence approach (later called as CND loop) of the Norwegian Armed Forces.

A cyber incident is: “the violation of an explicit or implied security policy” [17]. This definition is far reaching as it contains: “attempts to gain unauthorized

access to a system or its data; disruption or denial of services, the unauthorized use of a system for processing or storing data; changes to system hardware, firmware, or software characteristics without the owner's knowledge, instruction, or consent" [17]. Tools to gain unauthorized access vary from social engineering to malware, and threat vectors are exploited by combinations of human and machine delivery tools. Whether or not a cyber incident is launched from outside or inside of an organization's network, in both cases surveillance tools capable of detecting intrusion and extrusion are needed.

However, the performance of technical tools for intrusion and extrusion detection is particularly limited when the nature of the incident is unprecedented [5]. The ability of a tool to think critically and to draw and make counter-intuitive conclusions and decisions that are not anticipated by an adversary - initiating an intrusion - is limited. Therefore, humans in the loop are still required. Higher-level cognitions such as intuition, judgement, improvisation and perspective-taking are necessary and they are particularly superior in the creative ad-hoc development of problem-solving strategies that are unique to novel problem solving (e.g. [34,37]).

A good example of human critical thinking was revealed in the documentary *Zero-Days* [13]. Engineers at Symantec described how they worked on figuring out the STUXNET malware. They analysed the code, attempted to make sense of it individually and as a knowledge-building collaborative team. They shared ideas across domains and began to look for indicators beyond the code, focussing their attention on the geopolitical context in which the attack was taking place. Sensemaking of the malware and its purpose meant merging clues from the operator level to the strategic level. Doing this required non-linear thinking, and it appears the Symantec engineers applied this technique as a consequence of the events rather than as pre planned action. This way of thinking and problem solving could be taught and encouraged in order to assist detection and analysis teams to improve their performance and expedite attribution.

In today's digital society cyber incidents occur on a daily basis and are often handled by Computer Security Incident Response Teams, CSIRTs. The name of the group can vary, but the task of the groups is primarily the same: to prevent malicious network intrusions from achieving their goal [40]. This is often achieved through early detection, effective responses that prevent further damage or spread, and rapid system recovery. The Norwegian Armed Forces' Computer Network Defence Team is an example of an CSIRT.

Depending on the CSIRTs operational focus a cyber incident can be viewed as a technical issue - meaning the solution is a skill contained within in the field of computer engineering; or the incident is an organisational issue - meaning the problem needs to be addressed beyond the digital network and involves human capacities in the physical domain. These views can frame how the incident is handled rather than form responses.

From a purely technical perspective, handling a cyber incident is about restoring the systems to normal operation as fast as possible. From an organisational perspective, incident handling is about managing impacts on business

continuity, customer loyalty and internal/external reputation. As such, handling a significant cyber incident is technically and organizationally demanding. The CSIRT and the management need to see the incident from a shared perspective if the response is going to ensure greater levels of resilience at a technical and organisational level. This suggests that improving human understanding and decision making in relation to a cyber incident is critically important when conducting CND and incident response to combat the capability of cyber power to: “influence tangible and intangible assets through digital means” [26].

The paper reviews human skills and identifies possibilities for performance enhancement with a perspective on incident detection. The human skills identified by previous research and current Computer Engineering Curricula is discussed in Sect. 2. Section 3 analyses the Norwegian Armed Forces’ cyber defence approach and shows where and how the human is needed. Section 4 gives a summary of the research done by authors to empower the human factor and Sect. 5 discusses how the previously summarized research can strengthen the Norwegian cyber defence approach. Section 6 concludes the paper, and future work is presented in Sect. 7.

2 Related Work: Human Skills in Incident Detection

In order to handle cyber incidents with different severities successfully, a CSIRT needs to follow methodological frameworks such as Wide Area Situational Awareness (WASA) to: “...ensure dynamic prevention and response services” [1]. Frameworks such as WASA encourage modes of collaborative cognition [7] among human, as well as emphasizing the criticality of context-awareness in critical infrastructure protection. In practice this means a team with differing skill-sets finding common ground. This can be achieved by taking actions to identify where their individual capacities and cognitions overlap and complement each others to an extent that macrocognitive functions occur for improved performance. Most large CSIRTs comprise of technical staff (analysts fulfilling differing functions), managers, legal and law enforcement liaison, public relations personnel, technical writers, IT and system engineering personnel, and administrative personnel [40]. Accordingly, not all personnel need to have deep technical knowledge. A basic understanding of systems and network architectures, operating systems, network-protocols and domain name systems is enough to talk the same language [40].

For a CSIRT to achieve as close to optimal performance as possible in the hybrid environmental conditions described above, continual effort to develop the *understand function* is essential [35]. Each team member should continually take a critical view of the CSIRT purpose, role and goal in order to remain reflected, conscious and grounded in relation to the complex cyber-physical system of systems they operate within. This requires shared situation awareness as well as cognitive information processing resources such as attention, working memory, cognitive flexibility, metacognitive awareness and perspective taking [36].

When viewing cyber incidents as semi-, ill- and severely ill-structured problems with complex relations to operational success in for example; defence forces,

economic entities, governmental and non-governmental entities, then critical thinking and metacognition [41] as well as curiosity, tenacity, initiative, integrity [40] and insight [24] are all traits that should be taught and developed for better performance. These skills and capacities represent nonroutine thinking that can support complex problem solving, expert communication and applied knowledge in real world settings [43]. Inter and intra CSIRT information sharing and maintaining trustful relationships requires good communication techniques and time management skills. These are metacognitive awareness practices that can be supported by for example reflection [9]. The array of skills highlighted above that are required to be complementary in a CSIRT are also included in current Computer Engineering Curricula [2]. This curricula highlights communication skills such as writing, speaking and active listening skills. It also acknowledges the importance of teamwork skills and includes soft 'personal' skills such as common sense, the ability to deal with people, a positive flexible attitude, reading character traits, and the ability to sense and stimulate reactions. The curricula also emphasizes gaining and sharing experience and the ability to engage in life-long learning activities from a personal side, as well as allowing for continuous access to the education institution. This fosters a much more open, holistic and flexible way of thinking about competence, and development of key capacities and skills [44] as an ongoing process throughout life. The continuous interaction between institutions and people with a common goal of learning and development is the foundation of improved cooperation and collaboration. For some, the human skills discussed above are a natural part of the CND process. However this is not the case for all. In order to be more precise concerning the human factor in cyber defence, we now analyse the Norwegian Defences CND-loop, with a focus on the human role.

3 The human in the Norwegian Defence CND-loop

This section highlights the human in the Norwegian Armed Forces' Computer Network Defence Team's CND-loop [10]. The approach, originally influenced by Bejtlich [4], has been adjusted in recent years, however the human focus remains high, making it suitable as a case study for the purpose of this paper. The CND approach (CND-loop) is shown in Fig. 1.

The aim of the **planning and preparation** phase is to create a defensible infrastructure. This phase is heavily dependent upon humans as it relies on decisions related to making a sustainable and resilient defensible network; meaning [4,32] how the network is monitored, controlled, minimized, remains current, and ensuring all actions that take place inside the infrastructure are traceable.

A *monitored* information infrastructure is achieved by deploying sensors (e.g. IDS, SIEM) and implementing access logs at critical points. Equipment can be already in place for example in firewalls, anti-virus programs and system logs. They can also be deployable allowing for more dynamic operations. Traffic monitoring is a technical issue, but the physical placement of sensors is a human driven decision and is based on risk assessment. The Norwegian Team has a risk



Fig. 1. CND procedure loop in the Norwegian Defence Forces.

assessment methodology for operational security that is similar to the IOS/IEC 27000 family. Risks are evaluated based on the overlap of three different factors: critical assets, vulnerability testing and threat assessment.

A *controlled* infrastructure has a control mechanism that will hinder the adversary in maneuvering freely within the network such as; firewalls, VLAN-technology, proxy-solutions, systems for perimeter- and access control, authentication solutions etc. Honeypots can also be used to exercise control. Once again, the actual mechanisms are technical, but decision-making for application in the physical domain and in the cyber domain are for humans to decide.

The sum of all functions and services in the information infrastructure can be regarded as the *attack surface*, and specific vulnerabilities the *attack vector*. Due to the hybridity and convergence of multiple domains it is unlikely that all vulnerabilities can be removed. However, it is possible to reduce the attack surface, by removing potential attack vectors that are not necessary in your network, leading a *minimized* infrastructure. Which services are needed both in cyber domain (Office-tools, Email, Internet Browser etc.) and in physical domain (Front Desk, Internal Post, etc.), depends on the purpose of the network. In some networks end-users are allowed to decide by themselves, whilst in others, the decision is made higher in the command chain.

Since it is not possible to remove all vulnerabilities whilst ensuring critical functions and services remain available, it is important that services are updated at all times. An updated infrastructure is called a *current* infrastructure. Depending on the service in cyberspace, updating can vary from a fully automated solution to an end-user's responsibility. The updating process should not only happen in cyberspace, but also in the cyber-physical space of the cyber

domain. For example, old components and legacy systems need to be replaced with modern technology. However, this upgrading is seldom work that CND Teams or an end-user will do themselves.

For an incident to be *traceable* requires a system for handling log data. This requires advanced planning as many critical system logs, such as those generated by NAT-units (Network Address Translation) are not stored with the CSIRT itself. It is also possible to log physical access but again this is normally not a task given to a CND team.

Collecting indicators is a broad topic that includes both technical solutions and human participation. Indicators can range from; a user reporting that a pdf-file did not open properly, a suspicious email, an alarm triggered by anti-virus program or IDS, a tip-off from collaborating organizations, anomalies detected by an analyst monitoring a network. Automatic detection by anti-virus, IDS, SIEM etc., are valuable, but against targeted-attacks, they may fall short as the attacker is well aware of their limitations. To improve targeted attack detection the method called the cyber kill-chain [18] is used by the CSIRT in the Norwegian Cyber Defence. The kill-chain was designed as a process for countering cyber espionage and cyber attacks and it has seven phases: reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on target. A thorough understanding of the kill chain increases the likelihood of detecting an attack, as each phase will generate unique indicators. In addition, the collection of indicators from earlier attacks in each phase of the kill-chain will help the CND team detect attacks that reuse attack techniques. Understanding the kill-chain is important also for end-users, as the delivery phase often concerns the end-users of a system and hence they become ‘first-line’ defenders of their own system.

Correlation and analysis refers to establishing the consequences and intent of detected attacks. When an attack is detected late, in for example the ‘command and control’ phase, humans must lead the analysis. Not as obvious, but equally important, is to conduct thorough analysis on attack types when the detection happened earlier in the kill-chain. It may be tempting to think that no harm has been done and avoid applying human resources on this analysis. However, it is good practice to do the analysis to avoid leaving the network open for future attacks. Defenders should attempt to imagine what would have happened in later phases if the attack had not been detected. This critical thinking approach can give a tactical advantage to the defender, as the adversary will most likely have to take a complete new approach. Analysis and synthesis of all phases in the kill-chain can also be used for more long-term and deeper strategic analysis, i.e., to determine patterns and behaviors specific to each adversary. This can be used to evaluate capabilities, doctrines, objectives, limitations and ultimately their intentions.

Correlation of the attack indicators is often a technical task, as large amounts of data needs to be analyzed in a relatively short time frame. The human task in this phase is to start asking critical questions such as: has the attack type occurred before, and if so, what were the physical consequences and the effects

in cyberspace; are allied partners experiencing similar attacks, or have they experienced it before; have any assets been compromised and if yes, to what extent; how might the attack affect ongoing and future operations both inside and outside of cyberspace; is the attack type and attacker a relevant actor in the wider operational context; assessing how similar attacks can be avoided in the future.

Asking critical questions may not provide sufficient or complete answers. However it can lead to better results in the **incident handling** phase. At this stage any response will have to be appropriately measured against the type of attack and the context in which it occurs. This may well require the team make calculated assumptions based upon incomplete answers, existing plans and attack knowledge gained from earlier phases in the CND-loop. In a military operational context, the severity of the attack combined with any key factors that arise in corresponding domains all add to the framing of the cyber-physical environment in which the attack took place. This could demand more extreme responses from cyber defence teams such as accelerating through the CND-loop phases in order to ‘restore’ as top priority. Sometimes it is important to stop the attack at once, other times observation is more valuable. The key here is that organisations and their integrated systems of cyber governance should have procedures that allow them to reach a decision point, where they can choose to either stop or observe. It must be attack severity and context/situation that defines what is the right choice. In a military operational context the different courses of action that follow can include; deny, disrupt, degrade, deceive, destroy, etc. and they are an integral part of the current military operations.

In the **restoring and follow up** phases the network has to be re-established to a state that allows for own freedom of movement. This means strengthening network defenses to ensure future attacks are detected and handled as early as possible in the kill-chain.

In summary, the purpose of the human factor in all phases of the Norwegian Defence CND-loop is to connect information received from both cyber, social and physical domains by asking critical questions that demand reflected answers. It is also a human responsibility to make certain key operational decisions relating to for example strategic goals, risk management, and human resource management. This role is complexified in today’s digital society - that includes digital battlefields - critical decision-making often has larger and farther reaching consequences beyond digital and physical borders. A common factor in this environment is the influential role of cyberpower. The uncertainty surrounding cyberpower effects [26] mean quite often decisions have to be made faster, lower in the command chain and in a context where greater amounts of information may not necessarily equate to increased situational awareness or operational clarity [6]. This adds cognitive strain at both the tactical level and strategic level, and influences how education of the CND operators and leaders should be done. Learning to cope and govern within this modern context needs to be addressed. Reliance upon technical skills combined with the latest technology developments

may only be reinforcing the weakest link that is commonly referred to as: ‘the human’.

In the next section we discuss the research we have conducted to meet these challenges when educating the next generation of CND operators and leaders.

4 The Norwegian Defence Cyber Academy (NDCA): Human Factor Research

The NDCA educates cyber savvy military personnel for the entire Norwegian Armed Forces; including the Norwegian Cyber Defence. In order to better address the needs of the Norwegian Armed Forces regarding the human element in the Cyber Domain, research has been conducted and the results have been - and will continue to be - implemented into the NDCA educational platform. In this section we discuss the research conducted to help cyber cadets perform better in contemporary military operations.

Incident detection, whether it is detected by means of computer aided recognition or by human detection, is always an event that a designated person has to manage and cope with. Prior experience and well-grounded theory support making familiar events more recognizable. For example familiarity with tools and techniques used in a new incident, could make coping with novel situations easier. In the study of the use of cognitive strategies to aid coping in different challenging situations [14] we found that having predefined cognitive strategies - and knowing how to apply them - can support gaining control of a situation/s by planned conscious action; hence improving performance. In the paper, we studied ten coping strategies (see Fig. 2) introduced by the NDCA to support better performance among cyber officer cadets engaged in engineering and military tasks. As shown in Fig. 2 the strategies *situational understanding* and *accept the situation* were found to be primary strategies, i.e., applied first before, the other strategies could be implemented. The remaining strategies correlated positively with performance, which led us to rename them: performance strategies.

The performance (coping) strategies were further studied together with the effect of motivation and physical condition on NDCA students’ cognitive abilities in a Cyber Cognitive Task Performance test. The experiment was conducted during a two-week long military endurance exercise, which was both physically and mentally demanding. Results showed that the cyber operator performs better if the goal of the cyber task was explained to her/him [15]. The coping strategy; *ability to control* was shown to have a positive effect on performance in cyber tasks. This supports the well grounded research on how self-efficacy can contribute to mastery, control and the capability to influence situations [3]. Our study also investigated if the physical condition of the cyber students had a positive effect on performance. The results were not strong; but in cases where a person had been in a physically demanding environment for an extended period of time, and the task lacked a clear explanation, then better physical condition supported good performance.

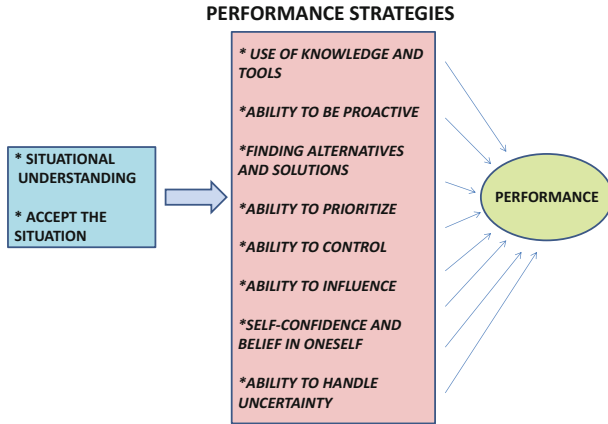


Fig. 2. Performance strategies [14].

The above study was strengthened when the students self-evaluated performance in the Cyber Cognitive Task Performance tests were correlated with their usage of the coping strategies [16]. The results showed that participants who reported using the *ability to control* and *self-confidence and belief in oneself* strategies performed significantly better. These results also support earlier research as these strategies contribute to self-efficacy; an individual's belief about his/her capabilities to complete a task with certain performance levels [3]. Situational self-efficacy has been shown to be a predictor for performance in different task domains [42], specifically in task-related performance [21]. Earlier research has also shown that in cases where cyber task demands and their relevance are defined, self-efficacy can enhance cyber oriented performance [8]. Together with this research, our findings suggest that motivating task-related information increases one's belief to control the situation, therefore supporting cyber task performance.

Further, our research showed that in situations where counterintuitive decisions are required, the combination of high *self-efficacy* and the high *availability of interoceptive information*, i.e., gut feelings, impairs decision-making success due to these factors increasing the probability that an easy but wrong decision is made when faced with complex problem solving [31]. These findings are particularly relevant when attempting to evaluate and improve performance in incident detection teams, as cyber incidents might have counterintuitive or powerful unanticipated effects that influence events beyond the primary intent.

In order to be able to perform well when managing a cyber incident the CSIRT must acquire more understanding about the complexity of the cyber incident and the wider situational context. The coping/performing strategies could be more usable, and self-efficacy higher, when a situation is familiar for a person. Incidents in the cyber domain are often complex and can vary in subtle and significant ways, but the common denominator for all incidents is they

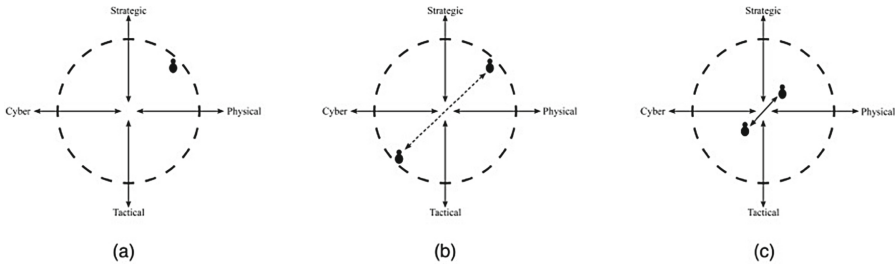


Fig. 3. OLB-model [27] as a procedure to communicate across the Hybrid Space [19].

connect the physical and cyber domain with human endeavour. To grasp the cyber-physical world connection better in a military context, the *Hybrid Space framework* (HS) [19] can be used. The HS defines the interconnection between the cyber domain and the physical domain. Simultaneously visualising these two aspects with tactical and strategic goals in decision-making and action can reveal tension between them, as well as potentially identifying and evaluating performance [28]. The framework is representative of today's digitized context and can support greater appreciation for the understand function in areas such as; the compression of command-levels in a future operating environment context [35]. The purpose of the HS is to open dialogical space for exploration and improved understanding for the competencies, human behavior and cognitive processes that should be in play to ensure that; no-matter-what the cyber situation, the team has the combined capacities and skills to cope better than before.

Individual understanding of the HS is not enough for collaborative efforts to work well in cyber operations. Horizontal and vertical mutual understanding among individuals is vital. This also needs to be complemented with the same ability to communicate horizontally and vertically inside and outside the collaborative environment. To strengthen the education of communication and understanding within a HS landscape, the *Orienting-Locating-Bridging model* (OLB-model) was introduced and applied [27]. The model is shown in Fig. 3 and depicts the process where communication partners - supported by metacognitive awareness - orientate themselves first, before attempting to accurately locate each other through perspective taking. These two first steps make bridging between communication partners less challenging, as grounding communication becomes more straightforward due to their ability to adapt communication content and style.

Improving the orienting phase of OLB can be achieved through developing metacognitive skills. This can help enable individuals to consciously know themselves and their environment well enough to safely proceed to locating others' cognition in either the same physical space, or different physical spaces facilitated by cyberspace. Initial testing of the hypothesis that higher metacognitive awareness would be positively associated with more movement in the HS has found that; a) metacognitive debugging strategies for cognitive regulation such as

accurate comprehension of performance and *self-regulation* could predict movement and b) evaluative and triggering metacognitive regulatory behaviours such as *searching for solutions* and *implementing new strategies* could also predict movement [28]. This preliminary empirical data might indicate that measuring and evaluating metacognitive processes could be a novel means of measuring and consequently improving performance in CND teams.

A further theoretical study looked at team cognition. Effective cyber security requires collaborative problem identifying and problem solving [20]. As shown in the STUXNET example (Sect. 1), the analyst had to work across domains and hierarchy, in a highly unstructured yet collaborative way. This illustrates a shift in formal structures towards more complex relations that challenges power-relations and modes of organisation and calls for a more adaptive approach to problem solving [33]. In the paper [20] we utilize the empirically grounded macrocognition research [22] and apply it in a CND team setting, to better understand the team cognition processes that occur in a HS context.

Recent research on teamwork demands on human performance has shown several social/team factors that can both facilitate and inhibit performance in the cyber domain. Analysts often spend a lot of time and effort into searching the web for information, which often can be held by other members of the team, and simple communication efforts as explained above, could provide the needed information [39]. This indicates that the analyst may actually contribute to heighten own cognitive load by adding information search into the process, in turn risking a decrease in communication and collaboration effort. Putting both the individual and team perspective together suggest that these are reciprocal cognitive processes influencing each other, and hence understanding team dynamics in a CND team could benefit from being unpacked in a top down approach in accordance with the naturalistic decision making paradigm [23]. Paying interest in both the phenomenon of teams in a cyber context and the inter and intra processes that support performance can be mutually beneficial [25]. We conducted studies on the influence of team workload demands in individual performance and found that communication and coordination demands significantly increased HS movements, while team workload timeshare demands and performance dissatisfaction impeded HS agility [30]. This supports the concepts presented in the OLB model that communication and coordination factors are needed to perform, while emotional aspects of teamwork (timeshare demand and performance dissatisfaction) and monitoring of teamwork could hinder the sharing of mental models explained by the OLB model.

5 Discussion

The CND loop is a normative model based on best practice. Seen in the context of the WASA framework it is comparable as we see that the six stages correspond to those specified in WASA. What differentiates the CND loop from the WASA framework is we argue that removing the human from any stage, and replacing him/her with automisation adds vulnerability due to an automatised

agents inability to adapt to dynamic contexts, that may require human sense-making capacities. The benefits from having a well-structured approach to a complex task is that it gives oversight and a path to unpacking an otherwise intangible problem. However, in cyber defence practice, each stage of the process requires individual and collaborative human input through cognitive processes like for example the need for sensemaking and decision-making for progress [20]. While the CND-process describes the necessary stages in which a successful cyber defence operation takes place, it is primarily cyber-technical focussed and assumes a purely rational human being providing input and making decisions. The planning and preparation phase requires long term strategic thinking and decision making in order to invest in the right type of technical equipment and the right type of human competence in order to ensure integrity, confidentiality and accessibility of cyber systems vital for the organisations purpose and mission. This requires mature leadership, proficient technicians and mutually inclusive communication that ensures common ground and understanding. Recent examples like WannaCry show that CND is not a 'thing' limited to the technical department, but intersects cyber and physical domains due to its potentially wide consequences. The paradox being that consequences could be mitigated by relatively easy measures involving humans updating systems. A common approach to the understanding of cyber and the dependability and impact of cyber incidents on the organisations mission is therefore required.

Expanding the view of cyber and making it an organisational priority denotes the importance of a common understanding of cyber and its impact on power relations and organisational structure. This also paves the way for a reduced distance between strategic leadership and CND-operator [19]. A consequence may be that good situation awareness in the cyber domain may not guarantee good overall performance, but it is a factor that can increase the likelihood of it [11]. In addition, better situational awareness of the physical environment and ongoing operations can add to overall performance. However, the general term 'situational awareness', empirically grounded in the physical world, may not be directly transferrable to the cyber domain [29]. When this is considered along with for example John Boyd's OODA loop - as a way to increase decision making through speed of thought and action [38] - then there are grounds for assuming that our research, focussing on the meta level of cognition, can support improved human performance in the CND-loop.

We believe that CND operators who have an advanced understanding and appreciation for the HS framework are more likely to find *planning and preparation* easier. This view is founded on the idea that higher levels of consciousness can support team and individual performance-optimisation through making better/more accurate judgements of own performance and appropriate initiation of change of cognition or action/behavior [30]. The ability to plan, monitor and evaluate one's own cognitive processes requires training in metacognitive awareness. Deeper contextual comprehension of the complex environment allows operators to appreciate where they are cognitively at any given time in a CND-loop. This idea is the central theme of cognitive agility as a method of evaluating

individual cognitive performance in cyberspace operations [28]. An explicit factor in cognitive agility is the need to enhance metacognitive awareness for better regulatory practices leading to improved performance. This is not particular to any phase of the CND-loop. Instead it is applicable to all, as - for now - every aspect requires human endeavour. Further, meta-awareness supports macrocognition for improved communication in socio-technical systems leading to better communication in the vertical plane of the HS [20, 28] and better team performance.

Application of coping strategies - in particular control - can support improved self-efficacy which our research has shown can support better performance in cyber tasks such as; *collecting indicators, correlation and analysis, and incident handling*. A CND teams' work is not so glamorous as movies tend to picture it. In fact, the phases; *collecting indicators and correlation and analysis*, sometimes involves long periods of tedious work that demands high levels of self-regulation. The motivation to endure can be strengthened if all team members are given information about ongoing military operations. Knowing the operation and understanding the importance of the CND teams work relating to the operation will increase the motivation of the team, as the perceived meaningfulness of the job has considerable implications for an employee's motivation [12].

Restoring and Follow-up are fundamentally human driven process. All members of a cyber defence team need to have a shared understanding of how the new defendable landscape will look. These two functions are closely related and rely heavily on the human's ability to - as far as possible - understand what has occurred. This means restoration and follow-up are appropriate in relation to the attacker's intent rather than simply getting services up and running again. This means giving appropriate levels of thought to the human process behind the attack in order to ensure these two phases mitigate future attacks. Understanding what future attacks might look like means thinking like the enemy and then bringing these intelligence led considerations into the planning and preparation phases. In essence, this means the *restoring and follow-up* phases have to be framed by how the incident is handled as a whole, rather than based on rapid actions to get systems up.

Investment in understanding how to improve a CND team can contribute to accelerated learning process, for example moving from novice to expert. Fundamentally though, better performance in the CND-loop will serve as a method of hardening the human in cyber-physical system. We claim that by implementing methods that can scaffold human cognition we will increase individual and team processes. This will improve situation awareness as a direct consequence of faster application of the CND-loop; as decision-making is founded upon better contextual clarity in a HS landscape.

6 Conclusions

In this paper we have focused on the human aspect in a cyber defence team by discussing the case of the Norwegian Cyber Defence approach. We have shown that the human has an important role in the process, not only by owning technical skills but also having the high-level cognitive skills that will help in critical

thinking, decision-making and communication. One needs to keep in mind that cyber defence is first and foremost about defending operations (business, military etc.) and only humans (so far) can understand what is important for operational success. We have also discussed the research results that can be used to empower human skills when educating the next generation of cyber defence operators.

Even though we have addressed the cyber defence team, learning lessons from and understanding how expert incident detection teams operate, could lead to earlier implementation of practices in basic education. With this evidenced based improved understanding, knowledge can be transferred and adapted to support wider societal non-expert first line defenders; the end-users of the systems. This way we have a chance to transform the weakest link to the strongest asset in cyber defence.

7 Future Work

The researchers intend to work collaboratively to gain more empirical research in order to build further on some initial findings. For example identifying how team workload factors can predict movement in the HS [30]. Data such as this helps validate the HS framework as it supports the idea that cognitive agility is the sum of conscious decision-making and conscious behaviour in a HS environment [28]. Assuming that all members of a CND team are operating with high levels of cognition they will be better able to plan, establish and operate a defensible network. Establishing if this is a result of combined skills and capacities leading to improved performance through increased metacognitive awareness and macrocognitive process is inspiring for the research group.

References

1. Alcaraz, C., Lopez, J.: Wide-area situational awareness for critical infrastructure protection. *Computer* **46**(4), 30–37 (2013)
2. Association for Computing Machinery: Computer Engineering Curricula 2016: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering. IEEE Computer Society, December 2016
3. Bandura, A.: *Self-efficacy: The Exercise of Control*. Freeman and Co., New York (1997)
4. Bejtlich, R.: *The Tao of Network Security Monitoring-beyond Intrusion Detection*. Addison-Wesley, Boston (2005)
5. Blumbergs, B., Pihelgas, M., Kont, M., Maennel, O., Vaarandi, R.: Creating and detecting IPv6 transition mechanism-based information exfiltration covert channels. In: Brumley, B.B., Rönning, J. (eds.) *NordSec 2016*. LNCS, vol. 10014, pp. 85–100. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47560-8_6
6. Buchler, N., Fitzhugh, S., Marusich, L., Ungvarsky, D., Lebiere, C., Gonzalez, C.: Mission command in the age of network-enabled operations: social network analysis of information sharing and situation awareness. *Front. Psychol.* **7**, 937 (2016)
7. Champion, M., Rajivan, P., Cooke, N., Jariwala, S.: Team-based cyber defence analysis. In: *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support* (2012)

8. Choi, M., Levy, Y., Hovav, A.: The role of user computer self-efficacy, cybersecurity countermeasures awareness, and cybersecurity skills influence on computer misuse. In: Pre-ICIS Workshop on Information Security and Privacy (2013)
9. Daudelin, M.W.: Learning from experience through reflection. *Organ. Dyn.* **24**(3), 36–48 (1996)
10. Dyrkolbotn, G.O.: Computer Network Defence in the Norwegian Armed Forces. NISlecture, January 2013. nislab.no/nislecture/nislecture_2013
11. Endsley, M.: Measurement of situation awareness in dynamic systems. *Hum. Factors* **37**(1), 65–84 (1995)
12. Gangé, M., Deci, E.: Self-determination theory and work motivation. *J. Organ. Behav.* **26**, 331–362 (2005)
13. Gibney, A.: Zero days. Documentary (2016)
14. Helkala, K., Knox, B., Jøsok, Ø.: How the application of coping strategies can empower learning. In: Proceedings of Frontiers in Education Conference. IEEE (2015)
15. Helkala, K., Knox, B., Jøsok, Ø., Knox, S., Lund, M.: Factors to affect improvement in cyber officer performance. *Inf. Comput. Secur.* **24**(2), 152–163 (2016)
16. Helkala, K., Knox, B., Jøsok, Ø., Lugo, R., Sütterlin, S.: How coping strategies influence cyber task performance in the hybrid space. In: Stephanidis, C. (ed.) HCI 2016. CCIS, vol. 617, pp. 192–196. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40548-3_32
17. Homeland Security, August 2016. www.dhs.gov/how-do-i/report-cyber-incidents
18. Hutchins, E.M., Cloppert, M.J., Amin, R.M., Lockheed Martin Corporation: White Paper: Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains (2011). www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf
19. Jøsok, Ø., Knox, B.J., Helkala, K., Lugo, R.G., Sütterlin, S., Ward, P.: Exploring the hybrid space. In: Schmorow, D.D.D., Fidopiastis, C.M.M. (eds.) AC 2016. LNCS (LNAI), vol. 9744, pp. 178–188. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39952-2_18
20. Jøsok, Ø., Knox, B.J., Helkala, K., Wilson, K., Sütterlin, S., Lugo, R.G., Ødegaard, T.: Macro-cognition applied to the hybrid space: team environment, functions and processes in cyber operations. In: Schmorow, D.D., Fidopiastis, C.M. (eds.) AC 2017. LNCS (LNAI), vol. 10285, pp. 486–500. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58625-0_35
21. Judge, T., Jackson, C., Shaw, J., Scott, B., Rich, B.: Self-efficacy and work-related performance: the integral role of individual differences. *J. Appl. Psychol.* **92**(1), 107–127 (2007)
22. Klein, D.E., Klein, H.A., Klein, G.: Macro-cognition: linking cognitive psychology and cognitive ergonomics. In: Proceedings of the 5th International Conference on Human Interactions with Complex Systems (2000)
23. Klein, G.: Naturalistic decision making. *J. Hum. Factors Ergon. Soc.* **50**(3), 456–460 (2008)
24. Klein, G.: Seeing what others don't, the remarkable ways we gain insight. *PublicAffairs* (2013)
25. Klein, G., Ross, K.G., Moon, B.M., Klein, D.E., Hoffman, R.R., Hollnagel, E.: Macro-cognition. *IEEE Intell. Syst.* **18**(3), 81–85 (2003)
26. Knox, B.J.: An exploration of the ways institutional development may be affected by the growing influence of cyberpower. Master's thesis. The Open University of the United Kingdom, Development Management Program, April 2017

27. Knox, B.J., Jøsok, Ø., Helkala, K., Khooshabeh, P., Ødegaard, T., Lugo, R.G., Sütterlin, S.: Socio-technical communication: the hybrid space and the OLB-model for science-based cyber education. *J. Mil. Psychol.* (2017, to appear)
28. Knox, B.J., Lugo, R.G., Jøsok, Ø., Helkala, K., Sütterlin, S.: Towards a cognitive agility index: the role of metacognition in human computer interaction. In: Stephanidis, C. (ed.) *HCI 2017. CCIS*, vol. 713, pp. 330–338. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58750-9_46
29. Kott, A., Wang, C., Erbacher, R.F.: *Cyber Defense and Situational Awareness*. Springer, Switzerland (2014). <https://doi.org/10.1007/978-3-319-11391-3>
30. Lugo, R.G., Kwei-Nahra, P., Jøsok, Ø., Knox, B.J., Helkala, K., Sütterlin, S.: Team workload demands influence on cyber detection performance. In: *Proceedings of 13th International Conference on Naturalistic Decision Making*, pp. 223–225. The University of Bath (2017). https://www.eventsforce.net/uob/media/uploaded/EVUOB/event_2/GoreWard_NDM13Proceedings_2017.pdf
31. Lugo, R.G., Sütterlin, S., Knox, B.J., Jøsok, Ø., Helkala, K., Lande, N.M.: The moderating influence of self-efficacy on interoceptive ability and counterintuitive decision making in officer cadets. *J. Mil. Stud.* **7**(1), 1–9 (2016)
32. Malmedal, B., *Cyberforsvaret: White Paper: Arkitektur for en Forsvarbar Informasjonsinfrastruktur* (2012). <https://norcydef.blogspot.no/2013/03/jeg-har-skrevet-et-whitepaper-om.html>
33. McChrystal, S., Collins, T., Silverman, D., Fussell, C.: *Teams of Teams: New Rules of Engagement for a Complex World*. Penguin, New York (2016)
34. Merza, M.: The importance of investing in people, September 2016. <http://federalnewsradio.com/commentary/2016/09/importance-investing-people/>
35. Ministry of Defence, United Kingdom: *Future trends programme future operating environment*, December 2015
36. Morrow, D.G., Fischer, U.M.: Communication in socio-technical systems. In: Lee, J.D., Kirlik, A. (eds.) *The Oxford Handbook of Cognitive Engineering*, pp. 178–199. Oxford University Press, Oxford (2013)
37. Murray, S.: Human skills are essential in battle against cyber crime, November 2016. <https://www.ft.com/content/46449768-7031-11e6-a0c9-1365ce54b926>
38. Osinga, F.: *Science, Strategy and War : The Strategic Theory of John Boyd*. Eburon Academic Publishers, Delft (2005)
39. Rajivan, P., Janssen, M.A., Cooke, N.J.: Agent-based model of a cyber security defence analyst team. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 57, pp. 314–318. SAGE (2013)
40. Ruefle, R., Dorofee, A., Mundie, D., Householder, A.D., Murray, M., Perl, S.J.: Computer security incident response team development and evolution. *IEEE Secur. Priv.* **12**(5), 16–26 (2014)
41. Smy, V., Cahillane, M., MacLean, P.: Cognitive and metacognitive prompting in ill-structured tasks: the art of asking. In: *Proceedings of International Conference on Information, Communication Technologies in Education* (2015)
42. Stajkovic, A., Luthans, F.: Self-efficacy and work-related performance: a metaanalysis. *Psychol. Bull.* **124**(2), 240 (1998)
43. The World Bank: *World development report 2016: digital dividends*, May 2016. <http://www.worldbank.org/en/publication/wdr2016>
44. Thomas, A.: What is development management? *J. Int. Dev.* **8**(1), 95–100 (1996)

CRBP-OpType: A Constrained Approximate Search Algorithm for Detecting Similar Attack Patterns

Ambika Shrestha Chitrakar^(✉) and Slobodan Petrović^(✉)

Norwegian University of Science and Technology (NTNU),
P.O. Box 191, 2802 Gjøvik, Norway
{ambika.chitrakar2,slobodan.petrovic}@ntnu.no

Abstract. Misuse-based Intrusion Detection Systems (IDS) have difficulties in detecting new attacks originating from the old ones, since they usually use exact search algorithms to detect attack patterns. To solve this problem, we can use approximate search. However, approximate search algorithms generate too many false positives. In this paper, we introduce constraints on the types of edit operations in the approximate search. Our algorithm applies an extension of the concept of Row-Wise Bit-Parallelism (RBP), which makes it highly efficient. The time complexity of the algorithm is at worst the same as that of the unconstrained approximate search algorithm based on the same RBP concept; otherwise, in most cases it is better than the unconstrained RBP. Experimental results show that our constrained approximate search algorithm produces a smaller number of false positives compared to the unconstrained RBP search algorithm, without reducing accuracy of search.

Keywords: Approximate search · Intrusion detection
Attack signatures · Constraints

1 Introduction

Misuse-based Intrusion Detection Systems (IDS) such as Snort [12] are used in practice to detect previously known attacks in networks. They use exact string matching of the known attack patterns in the network traffic. Such systems fail to detect unknown attack patterns even if they are very similar to the known ones. Approximate search algorithms are capable of matching similar strings and as such they can be used to detect unknown attack patterns similar to the known ones, up to a pre-defined level of tolerance k . However, they generate too many false positives. Constrained approximate search, which is a new sub-domain of approximate search algorithms, has ability to reduce the number of false positives by defining constraints on the edit operations [6, 7].

In this paper, we present a new constrained approximate search algorithm, CRBP-OpType. This algorithm applies an extension of the concept of Row-Wise Bit-parallelism (RBP) [15] that stands behind a family of unconstrained

and constrained approximate search algorithms and ensures their high efficiency. The CRBP-OpType algorithm allows constraints on the types of edit operations (insertions, deletions, and substitutions) that we want to use in the approximate search. By limiting the use of certain types of edit operations, we improve the performance of the algorithm and at the same time we enable usage of a-priori knowledge obtained by threat intelligence to reduce the number of false positives.

The application scenario of the CRBP-OpType algorithm can be the following: Suppose our threat intelligence obtained information about the way the potential attacker changes the attack patterns in order to bypass an IDS. This information can be in the form of the types of edit operations he/she uses. We can incorporate these types as constraints in our search algorithm. In such a way, we filter out the impossible transformations of the original attack pattern, which results in reducing the number of false positives.

We performed an experiment to demonstrate the applicability of the CRBP-OpType constrained approximate search algorithm to detect similar attack patterns. We used previously known SQL injection attack patterns as search patterns and randomly generated strings similar to these attack patterns. We then simulated CRBP-OpType and unconstrained RBP [15] to detect the attack patterns from the list of their corresponding search strings. The experimental results show that the CRBP-OpType efficiency is in most cases better than that of the unconstrained RBP approximate search algorithm and that it generates a smaller number of false positives compared to the unconstrained RBP. The algorithms are not limited to detection of SQL injection attack patterns only: they can be applied to detect any new attacks that are similar to the known attacks.

The structure of the paper is the following: Sect. 2 provides background on the concepts of approximate search and bit-parallelism and reviews the related work. The CRBP-OpType constrained approximate search algorithm is presented in Sect. 3. Section 4 presents the results of the experimental work. It is followed by the discussion in Sect. 5. Section 6 concludes the paper.

2 Background and Related Work

IDS heavily relies on exact string matching and its overall performance depends on the performance of the search algorithm that it uses. SNORT for example uses Aho-Corasick [2] multi-pattern search engine. In past few years, a lot of improvement on original Aho-Corasick algorithm have been done including hardware approaches to accelerate the efficiency of the algorithm [1,3,4]. However, these algorithms do not solve the problem of detecting unknown attacks that are similar to the known ones.

This section provides basic concepts about our new constrained search algorithm, which is capable of detecting similar attack patterns. It introduces a new classification of the approximate search algorithms, based on the use of edit operations. Since both the CRBP-OpType and unconstrained RBP [15] exploit the bit-parallelism technique, a brief explanation of how the approximate search algorithms in general exploit bit-parallelism is also provided.

2.1 Classification of Approximate Search

Search algorithms use distance functions such as Hamming distance [13], Levenshtein distance (also known as edit distance) [9], and q-gram distance [14] to match strings. Approximate search algorithms usually use Levenshtein distance. They match the search pattern (P) in the search string (T) by allowing certain level of error on the edit operations such as character insertions, deletions, and substitutions.

We classify approximate search algorithms into two groups: unconstrained and constrained, based on the use of edit operations. In unconstrained approximate search, we define only maximum number of allowed errors (k) on the edit operations, see for example [10]. The constrained approximate search is a sub-domain of the approximate search, where a-priori knowledge about the possible types of errors can be applied on edit operations. An example of a constrained approximate search algorithm is a pair of algorithms Sankoff-Indels and CRBP-Indels [6]. These algorithms apply constraints on the maximum number of indels, where by indels we assume insertions and deletions counted together. Another constrained approximate search algorithm is CRBP-OpCount [7], where constraints on the maximum number of each edit operation are defined.

2.2 Approximate Search and Bit-Parallelism

Bit-parallelism technique has been studied for the last 25 years in the field of string matching, see for example [8, 11]. The advantage of using bit-parallelism is in reduction of the number of operations by a factor of the number of bits of a computer word (w), which helps to improve the speed of an algorithm. Row-Wise Bit-Parallelism (RBP) [15] and Diagonal-Wise Bit-Parallelism (DBP) [5] have been used in some of the unconstrained approximate search algorithms that exploit this technique. In case of constrained approximate search, bit-parallelism has been applied in CRBP-Indels [6], and CRBP-OpCount [7]. However, the space complexity of these algorithms can be a limiting factor of their practical application, since they use counters to keep track of the numbers of particular edit operations.

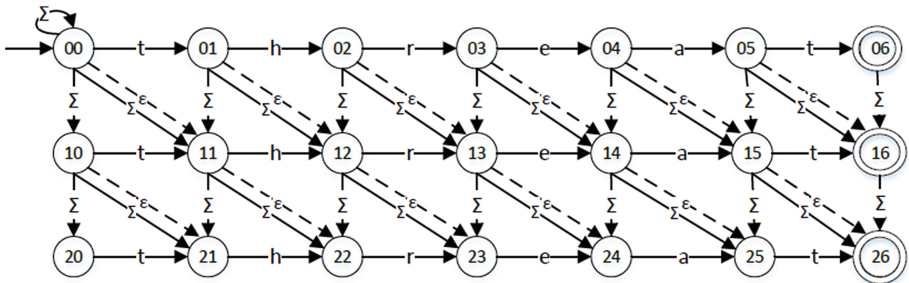


Fig. 1. An NFA for the search pattern “threat”, permitting up to 2 errors

Approximate search algorithms that use bit-parallelism technique simulate a non-deterministic finite automaton (NFA) for a given search pattern and then a search string is fed into it to try to find a match. Figure 1 shows an NFA assigned to the search pattern “threat” that allows up to 2 errors. Each row represents matching with the row number of allowed errors (0^{th} row represents matching with no errors, 1^{st} row with 1 error etc.). Each node/state in the Fig. 1 is labeled with a row number (the first index) and a node number (the second index) in that row. This is done to represent the states in an array form. A node with a self-loop is the initial state and the nodes with double-circles are the final states. Arrows represent the transitions from one state to another. A horizontal transition represents a character match, a vertical transition represents a character insertion, a dashed-diagonal transition represents a character deletion, and a solid-diagonal transition represents a character substitution. The substitution and the insertion characters are from the alphabet Σ . The symbol ϵ denotes so-called ϵ -transitions, which are the transitions that do not consume any input characters; in this case, deletions are modelled with ϵ -transitions. We say there is an occurrence of a search pattern at a certain character position of the search string, if any of the final states is active at that point. An active state is a state, which can be reached with a successful transition.

When simulating the NFA using the bit-parallelism technique, each node is assigned an activity bit (0 or 1). 1 represents an active state and 0 represents an inactive state. Each transition in the NFA for an input search symbol is simulated by using an update formula. These update formulas consist of operations such as AND (&), OR (|), and SHIFT-LEFT (<<) on the bits of the NFA. The concrete update formula depends on the search algorithm. For example, unconstrained RBP [15] simulates NFA transitions in a row-wise fashion, whereas unconstrained DBP [5] simulates them in a diagonal-wise fashion. We get bits for each row of the NFA after using the update formula for a particular character of the search string as input.

3 CRBP-OpType: A Constrained Approximate Search

CRBP-OpType constrained approximate search algorithm is based on the concept of Row-Wise Bit-Parallelism (RBP) [15]. Similar to the unconstrained RBP, it exploits the bit-parallelism technique by simulating a non-deterministic finite automaton (NFA) for a given search pattern and uses the update formula to perform the search in the given search string. The update formula used in the algorithm is given in the Eqs. (1) and (2) [6, 15].

$$R'_0 \leftarrow ((R_0 \ll 1) | 0^{m-1} 1) \& B[t_j] \quad (1)$$

$$R'_i \leftarrow ((R_i \ll 1) \& B[t_j]) | R_{i-1} | (R_{i-1} \ll 1) | (R'_{i-1} \ll 1) | 1 \quad (2)$$

This update formula follows a row-wise fashion, simulating transitions for each row in parallel. The first part of the update formula (the Eq. (1)) computes the horizontal transitions only for the 0^{th} row (R'_0) of the NFA for a particular

input search symbol t_j , where j is a position of the input search symbol in the search string. The second part of the update formula (the Eq. (2)) computes all the transitions (horizontal, vertical, solid-diagonal, and dashed-diagonal, respectively) for the i^{th} row (R'_i where, $i = 1, \dots, k$) of the NFA for the search string symbol t_j . In the Eq. 2, there are four parts that are mutually ORed. The first part corresponds to a match with a character t_j , the second part corresponds to an insertion, the third part corresponds to a substitution, and the fourth part corresponds to a deletion.

$B[t_j]$ in the update formula is a bit-mask for a particular character t_j of the search string at position j . The bit-mask has to be generated for all the characters that are contained in the search pattern in advance. As an example, let us assume that we have a search pattern “cocoa” and we use English alphabet (Σ) in both the search pattern and the search string. In this case, for the characters that are not available in the search pattern, all the bits of the corresponding bit masks are set to 0. For each unique character of the search pattern, we traverse the search pattern from left to right and then set the bits to 1 in the positions of the search pattern where that character is located. We reverse the bit sequences and store them as bit-masks for those particular unique characters. In our example, for the search pattern “cocoa”, we get the bit-masks as follows:

$B[c] = 00101$, $B[o]: 01010$, $B[a]: 10000$, $B[*]: 00000$.

Here, $B[*]$ are the bit-masks for all the remaining characters that are not found in the search pattern. R_0 and R_i in the update formula hold all the bits of the 0^{th} row and the i^{th} row, respectively, for the search symbol at position $j - 1$. R'_{i-1} holds all the bits of the $(i - 1)^{st}$ row for the search symbol at position j . 1 is ORed at the end of the formula. This is because we assign 1 instead of 0 to the shifted bit after a SHIFT-LEFT operation.

The CRBP-OpType algorithm that simulates the NFA (see Fig. 1) in an array format is presented in the Subsect. 3.2. Important variables that are used in the CRBP-OpType algorithm are explained in Subsect. 3.1.

3.1 Variables for the CRBP-OpType

The variables used in the CRBP-OpType constrained approximate search algorithm are listed in Table 1.

In Table 1, P is a search pattern of length m and T is a search string of length n . k is the maximum number of allowed errors for the approximate matching. $B = [char \rightarrow bitmask]$ is a bit-mask for all the characters that can be applicable in the search. This variable is an array, which has a character as an array index and value as the character’s bit-mask. row is an array of rows with its bit sequence. The number of rows is equal to the number of allowed errors $(k) + 1$. pos is a position of the current input search symbol in the search string and $tpos$ is the character at that position. $newR$ holds a sequence of bits of the row i of the NFA, for which the update formula is being computed. $oldR$ consists of a sequence of bits in the row $i - 1$ of the NFA for the search character at $pos - 1$.

Table 1. Variables used in the CRBP-OpType (see the explanation below)

$P = p_1p_2 \dots p_m$ - search pattern	$T = t_1t_2 \dots t_n$ - search string
k - maximum number of allowed errors	$B = [char \rightarrow bitmask]$
$row = [r_0, r_1, \dots, r_k]$ - array of rows	pos - current search character position
$tpos$ - character at pos	$oldR$ - bits in (i-i) row for the char. at $pos-1$
$newR$ - bits in the row i	$allowI$ - flag for insertions (I)
$allowE$ - flag for deletions (E)	$allowS$ - flag for substitutions (S)
$newRR$ - temporary $newR$	

$allowI$, $allowE$, and $allowS$ are the flags that indicate if insertions, deletions, and substitutions, respectively, are allowed or not. The transitions are allowed if their corresponding flag variables are set to *true*.

3.2 Algorithm: CRBP-OpType

The CRBP-OpType algorithm uses a-priori knowledge on the possibility of using the type of edit operations to solve the search problem. This algorithm is capable of limiting the constraints on the type of edit operations and it avoids unnecessary computation of the errors. For example, if we allow up to 2 substitutions to find the search pattern “threat” in the search string then the NFA from the Fig. 1 will look like the one presented in Fig. 2. This NFA does not contain transitions for insertions and deletions and these edit operations are not computed in the second part of the update formula (the Eq. (2)).

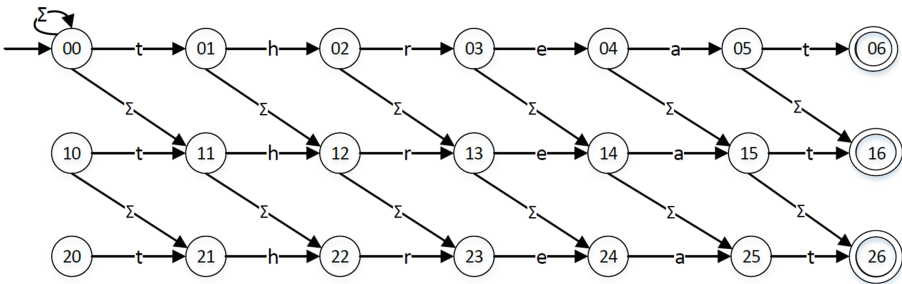


Fig. 2. An NFA for the search pattern “threat”, permitting up to 2 substitutions

Allowing only substitutions (S), only deletions (E), only insertions (I), allowing any combination of substitutions and insertions (IS), allowing any combination of deletions and substitutions (ES), and allowing any combination of insertions and deletions (IE) are the possible constraints in the CRBP-OpType algorithm.

Equation 3 shows a generic formula to compute the number K of possible combinations of edit operations for the approximate search algorithms. Here, k is the maximum number of allowed errors and N is the number of types of errors used in the algorithm. The number of possible combinations of errors depends on k and N . The smaller the values of k and N , the lesser will be the number of possible combinations of errors. A smaller number of combinations of errors also increases the performance of the algorithms, since each error type has certain operational cost.

$$K = \sum_{l=1}^k \binom{N+l-1}{l} \quad (3)$$

When the CRBP-OpType allows all the three edit operations in its constraint set (IES), the algorithm works exactly the same way as the unconstrained RBP. In this case, both the CRBP-OpType and the unconstrained RBP will match similar strings with the same possible combinations of errors. For example, let us assume that we want to apply the CRBP-OpType search algorithm to detect the search pattern with maximum 2 errors ($k = 2$) by allowing all the edit operations ($N = 3$). In this case, when we apply formula in the Eq. 3, the CRBP-OpType algorithm can match the search pattern in the search string with 9 possible combinations of errors, which is also the same in the case of the unconstrained RBP. These possible error combinations are listed below:

- 1 insertion,
- 1 deletion,
- 1 substitution,
- 2 insertions,
- 2 deletions,
- 2 substitutions,
- 1 insertion and 1 deletion,
- 1 insertion and 1 substitution, and
- 1 deletion and 1 substitution

The CRBP-OpType algorithm differs from the unconstrained RBP if we limit the use of certain edit operations, which is not possible in case of unconstrained RBP. Limiting the types of edit operations in the approximate search reduces the number of possible combinations of errors and at the same time increases the performance of the algorithm. For example, if a constraint on the combination of insertions and substitutions (IS) edit operations ($N = 2$) is set with $k = 2$ then the CRBP-OpType algorithm will try to match the similar strings with 5 different possibilities of errors (see Eq. (3)):

- 1 insertion,
- 1 deletion,
- 2 insertions,
- 2 deletions, and
- 1 insertion and 1 deletion.

However, the unconstrained RBP will still consider 9 possible combinations of 3 edit operations ($k = 2$ and $N = 3$). The performance of the CRBP-OpType algorithm can be further improved if we use only 1 type of edit operation as its constraint. In this case, the CRBP-OpType tries to match strings with only k possibilities of errors. For example, if we define constraint on only substitutions ($N = 1$) with up to 2 errors ($k = 2$) then the CRBP-OpType algorithm will match the strings with 2 possible error types only:

- 1 substitution, and
- 2 substitutions.

The theoretical worst-case (when $N = 3$) time complexity of this algorithm is $O(k[m/w]n)$. When $m \leq w$, it becomes $O(kn)$. Here, n is the length of the search string, m is the length of the search patten, and w is the number of bits in a computer word. This NFA simulation cost is the same as in the unconstrained RBP [15]. However, by limiting the use of the edit operations, the CRBP-OpType reduces the unnecessary computation of certain edit operations and speedup its performance over the unconstrained RBP.

The CRBP-OpType constrained approximate search algorithm has three parts: defining bit-masks for the characters, initializing the NFA assigned to the search pattern, and performing the search. Pseudocodes for these parts are given in Algorithm 1 and its key variables are listed in the Subsect. 3.1. Note that we count bits from the right (LSB = 1, MSB = m) in the algorithm.

Algorithm 1. CRBP-OpType

```

1: procedure CRBP-OpType( $P, T, allowI, allowE, allowS, k$ )
2: Defining Bit Masks:
3:   for  $c \in \Sigma$  do  $B[c] \leftarrow 0^m$            //  $\Sigma =$  alphabets
4:   for  $i \in 1 \dots m$  do  $B[P[i]] \leftarrow B[P[i]] | 0^{m-i}10^{i-1}$ 
5: Initializing the NFA:
6:   if  $allowE = true$  then
7:     for  $i \in 0 \dots k$  do  $row[i] \leftarrow 0^{m-i}1^i$ 
8:   else
9:     for  $i \in 0 \dots k$  do  $row[i] \leftarrow 0^m$ 
10: Applying the Search:
11:   for  $pos \in 1 \dots n$  do
12:      $oldR \leftarrow row[0], newR \leftarrow (oldR \ll 1) \& B[tpos], row[0] \leftarrow newR$ 
13:     for  $i \in 1 \dots k$  do
14:        $newRR \leftarrow (row[i] \ll 1) \& B[tpos]$ 
15:       if  $allowI = true$  then  $newRR \leftarrow newRR | oldR$ 
16:       if  $allowE = true$  then  $newRR \leftarrow newRR | (newR \ll 1) | 1$ 
17:       if  $allowS = true$  then  $newRR \leftarrow newRR | (oldR \ll 1) | 1$ 
18:        $newR \leftarrow newRR, oldR \leftarrow row[i], row[i] \leftarrow newR$ 
19:       if  $newR \& 10^{m-1} \neq 0^m$  then then report an occurrence at pos

```

Defining Bit Masks: Bit masks are assigned to all the characters that are contained in the search pattern. This process is the same as the bit-mask generation process of the unconstrained RBP [15].

Initializing the NFA: In this step, all the bits for $k+1$ rows of the NFA should be initialized. In case of the unconstrained RBP [15], i consecutive bits (starting from 1) from right to left are set to 1, where i is the number of rows in the NFA. For example, if $k = 2$ for the search pattern “threat” then the initialization of the rows in the NFA is the following:

`row[0] = 000000, row[1] = 000001, row[2] = 000011`

In case of CRBP-OpType algorithm, this process is followed only if deletions are allowed. Otherwise, all the bits in each row of the NFA are set to 0. This is because only deletion transition is possible from the current input search character ($tpos$) at position pos of the search string; substitutions and insertions are performed from the search character at position $pos-1$. Note that the number of bits in each row is equal to the length of the search pattern m .

Applying the Search: Every input search symbol is passed through the NFA and bits for each row are calculated by using the update formula (see Eqs. (1), and (2)). The first part of the update formula for the CRBP-OpType remains the same as in the unconstrained RBP [15]. The second part involving all the edit operations contains the formula to update the rest of the NFA rows. Since we can limit the use of edit operations in the CRBP-OpType algorithm, the constraint on the type of edit operations should be checked before applying the formula for them.

In case of CRBP-OpType, the second part of the update formula (2) is partitioned and the allowed transitions are checked from the list of pre-defined constraints. This algorithm includes a transition formula only if that transition is allowed in the constraint. For example, if we allow only substitutions ($allowS = true$) and deletions ($allowE = true$) to find the occurrences of the search pattern, then we do not calculate transitions for the insertions. A match is reported when the last bit of any row is active.

4 Experimental Results

In our experiment, we have used previously known signatures from the SQL injection attacks and we considered them as search patterns. The selected attack signatures were `1=1` and `'1'='1'`. These examples can be found in the “sql.rules” file of the Snort IDS. SQL injection attack signatures are relevant in our experiment, since the same attack can be performed by using multiple similar attack signatures. We randomly generated similar patterns from each of the selected attack signatures, which we considered search strings.

An SQL injection code below shows how one of these attack signatures can be used in an attack. The SQL code fetches all the users from the users database without checking any valid user credentials. This SQL statement is valid even with a blank username because `1=1` is a valid arithmetic statement and because `--` comments everything that appears after it.

```
SELECT * FROM users WHERE username='' OR 1=1-- and password=''
```

We assumed that the attacker could exploit `1=1` search pattern by using up to 2 substitutions. This is due to the necessity to replace numbers on the left and the right sides of the equal sign, if any of them is changed. By allowing maximum 2 substitution errors, we randomly generated 200 search strings for `1=1`, out of which, only 10 were the valid attack patterns. Some of the randomly created search patterns include `2=2`, `3=3`, `1=2`, and `2=3`.

For the search pattern `'1'='1'`, we assumed that the attacker can perform maximum 2 character insertions and 2 character substitutions on it. By allowing maximum 2 character insertions and 2 character substitutions, we randomly generated 1800 similar search strings. Out of 1800 search strings, only 180 were the valid attack patterns for the SQL injection. Some of these random strings include `'2'='2'`, `'12'='12'`, `'a'='a'`, and `'12'='2'`.

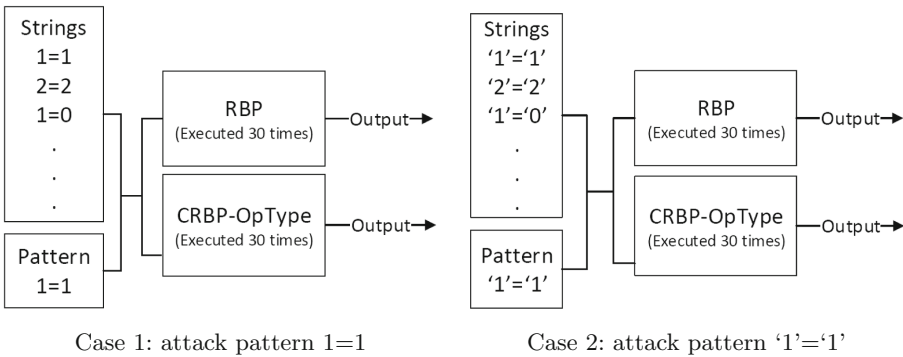


Fig. 3. Applying approximate search algorithms for selected SQL injection patterns

We studied two cases in the experiment. In the first case, we had an attack pattern `1=1` and 200 search strings. In the second case, we had an attack pattern `'1'='1'` and 1800 search strings. We simulated Row-Wise Bit-Parallel (RBP) unconstrained approximate search, and CRBP-OpType constrained approximate search algorithm in both cases in order to find the occurrences of the attack pattern in their corresponding lists of search strings. In order to measure the speed of these algorithms, we executed both of them 30 times and computed their average CPU time consumption. The process of the experiment for both cases is depicted in Fig. 3. The simulation of these algorithms is performed in a quad-core, 2.7 GHz processor computer with 8 GB RAM.

In the first case of the experiment, we set $k = 2$ for the unconstrained RBP, and the tolerance of maximum 2 substitutions for the CRBP-OpType. In the second case, we set $k = 4$ for the unconstrained RBP, and $k = 4$ with substitutions and insertions (IS) as constraints for the CRBP-OpType. In the case of the unconstrained RBP, the errors can include any combination of edit operations (insertions, deletions, and substitutions), whereas in the case of the CRBP-OpType, it can include any combination of insertions and substitutions only (see Subsect. 3.2 for more details). The number of false positives obtained by executing the simulated algorithms for both cases are given in Table 2.

Table 2. False positive results from the CRBP-OpType and the RBP

Patterns	Strings	Algorithms	Tolerance	TP	FP	TN
1 = 1	200	RBP	2	10	181	9
		CRBP-OpType	k=2, S	10	159	31
'1' = '1'	1800	RBP	4	182	1606	12
		CRBP-OpType	k=4, IS	182	1581	37

In Table 2, TP indicates the total number of true positives, FP indicates the total number of false positives, and TN indicates the total number of true negatives obtained by each approximate search algorithm. In our case, true positives are the matched search strings, which still ensure successful SQL injection.

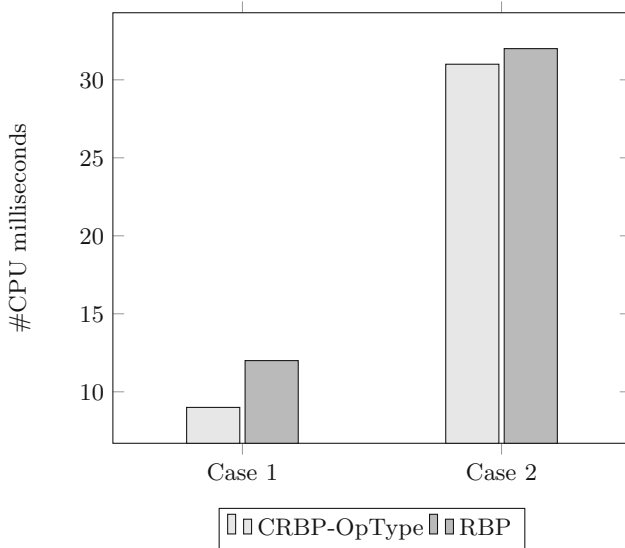


Fig. 4. Speed comparison of the CRBP-OpType and the unconstrained RBP

False positives are the matched strings, which lead to unsuccessful attempts of SQL injection, and true negatives are the strings, which are not matched with the attack pattern and at the same time do not give a valid SQL injection. We can see that in both cases, CRBP-OpType has generated fewer false positives compared to the unconstrained RBP algorithm.

Figure 4 compares the speed of the CRBP-OpType constrained approximate search algorithm and unconstrained RBP approximate search algorithm. In both cases, CRBP-OpType has outperformed the RBP. This result was expected (see Subsect. 3.2) as the CRBP-OpType algorithm was limiting the use of edit operations, whereas the unconstrained RBP algorithm used all the edit operations in both cases of the experiment. The number of used CPU milliseconds will vary with the use of experimental environment such as hardware or CPU, platform and programming language, way of implementation etc.

5 Discussion and Future Work

It is recommended to apply security measures in the software codes to prevent them from the SQL injection like attacks. However, it is enough for an attacker to find a single software fault to perform an attack. Therefore, without completely depending on the programmers, it is desirable to defend the systems against the attacks. Intrusion Detection Systems (IDS) have capability of detecting known attacks. However, they fail to detect zero-day attacks. They cannot even detect the unknown attacks that are similar to the known ones. This is because most of the IDS use exact search to find the intrusions in the network traffic.

Our experimental results (see Sect. 4) shows the possibility of using both the constrained and unconstrained approximate search algorithms to detect new attacks similar to the known ones. However, the unconstrained RBP approximate search generated more false positives compared to the constrained approximate search algorithm CRBP-OpType. This is because the CRBP-OpType allows us to be more precise in selecting the type of edit operations if we acquire intelligence in advance (for example, by using other sources of side information, such as threat intelligence data from various forums etc.), which is not possible in the case of the unconstrained RBP. In the first case of the experiment with the search pattern $1=1$, the unconstrained RBP looked for all the possible insertions, deletions, and substitutions on the search pattern with maximum 2 errors, whereas the CRBP-OpType looked for maximum 2 substitutions only. Similarly, in the second case where the search pattern was $'1'='1'$, the unconstrained RBP looked for all the possible combinations of the edit operations, whereas the CRBP-OpType looked for the possible combinations of insertions and deletions only.

The CRBP-OpType constrained approximate search algorithm performed better than the unconstrained RBP regarding the execution speed as well. The reason is the same, the CRBP-OpType checks for fewer edit operations than the unconstrained RBP. In the first case, the CRBP-OpType used only substitutions and in the second case, it used only 2 edit operations. The unconstrained RBP on

the other hand used all the possible combinations of 3 edit operations. Since the CRBP-OpType used two edit operations (insertions = 2, and substitutions = 2) in the second case, the speed difference between the CRBP-OpType and the unconstrained RBP is smaller in the second case than their speed difference in the first case.

In the future, we plan to consider a real case scenario in the experiment, where we will first demonstrate the fact that intrusion detection systems are not able to detect attacks whose attack signatures are not available in their database. Then we will apply all the constrained approximate search algorithms to detect similar attack patterns that were not detected by the classical intrusion detection system. We will also compare their results and performance with the known unconstrained approximate search algorithms.

6 Conclusion

The paper proposed a new constrained approximate search algorithm called CRBP-OpType to detect the unknown attack signatures that are similar to the known ones. This algorithm allows one to apply a-priori knowledge about the possible type of edit operations required to solve the search problem. A constraint on the type of edit operations can be defined in the CRBP-OpType algorithm while performing the approximate search. The worst-case time complexity of this algorithm is $O(kn)$, which is the same as with the unconstrained RBP approximate search algorithm. Here, k is the maximum allowed number of errors, and n is the length of the search string.

An experiment was performed to show how the CRBP-OpType constrained approximate search algorithm could be beneficial in detection of the similar attack patterns, compared to the unconstrained RBP approximate search algorithm. The experimental results show that although the CRBP-OpType algorithm has the same worst-case time complexity as the unconstrained RBP, it can speed-up the performance if fewer edit operation types are used as constraints in the algorithm. The experimental results also show that the CRBP-OpType algorithm can reduce the number of false positives compared to the unconstrained RBP algorithm.

References

1. Norton, M.: Optimizing Pattern Matching for Intrusion Detection. <https://snort.org/documents>. Accessed 2017
2. Aho, A.V., Corasick, M.J.: Efficient string matching: an aid to bibliographic search. *Commun. ACM* **18**(6), 333–340 (1975)
3. Lin, C.H., Liu, C.H., Chien, L.S., Chang, S.C.: Accelerating pattern matching using a novel parallel algorithm on GPUs. *IEEE Trans. Comput.* **62**(10), 1906–1916 (2013)
4. Yoon, J., Choi, K.I., Kim, H.: A memory accessing method for the parallel Aho-Corasick algorithm on GPU. In: 2016 International Conference on Information Science and Security (ICISS), pp. 1–3 (2016)

5. Baeza-yates, R., Navarro, G.: Faster approximate string matching. *Algorithmica* **23**, 127–158 (1999)
6. Chitrakar, A.S., Petrovic, S.: Approximate search with constraints on indels with application in SPAM filtering. In: Norsk Informasjonssikkerhetskonferanse (NISK) (2015)
7. Chitrakar, A.S., Petrovic, S.: Constrained row-based bit-parallel search in intrusion detection. In: Norsk Informasjonssikkerhetskonferanse (NISK) (2016)
8. Faro, S., Lecroq, T.: Twenty years of bit-parallelism in string matching. In: Festschrift for Bořivoj Melichar, pp. 72–101 (2012)
9. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys.-Dokl.* **10**(8), 707–710 (1966). Translated from *Dokl. Akad. Nauk SSSR* **163**(4), 845–848 (1965)
10. Navarro, G.: A guided tour to approximate string matching. *ACM Comput. Surv.* **33**(1), 31–88 (2001)
11. Navarro, G., Raffinot, M.: *Flexible Pattern Matching in Strings: Practical Online Search Algorithms for Texts and Biological Sequences*. Cambridge University Press, New York (2002)
12. Martin., R.: Snort - lightweight intrusion detection for networks. In: Proceedings of the 13th USENIX Conference on System Administration, pp. 229–238 (1999)
13. Sankoff, D., Kruskal, J.B.: *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison Wesley, Reading (1983)
14. Ukkonen, E.: Approximate string-matching with Q-grams and maximal matches. *Theor. Comput. Sci.* **92**(1), 191–211 (1992)
15. Wu, S., Manber, U.: Fast text searching allowing errors. *Commun. ACM* **35**(10), 83–91 (1992)

Multistage Downstream Attack Detection in a Cyber Physical System

Rizwan Qadeer¹(✉), Carlos Murguia¹, Chuadhry Mujeeb Ahmed¹,
and Justin Ruths²

¹ Singapore University of Technology and Design, Singapore, Singapore
{rizwan.qadeer,murguia_rendon}@sutd.edu.sg, chuadhry@mymail.sutd.edu.sg

² University of Texas Dallas, Richardson, USA
jruths@utdallas.edu

Abstract. We present an attack detection scheme for a water treatment system. We leverage the connectivity of two stages of the process to detect attacks downstream from the point of attack. Based on a mathematical model of the process, carefully crafted and executed attacks, are detected by deploying CUSUM and Bad-Data detectors. Extensive experiments are carried out and the results show the performance of the proposed scheme.

Keywords: CPS · CPS security · ICS security
Water treatment systems

1 Introduction

Cyber Physical Systems (CPS) are the integration of computing elements with the physical world [8]. The incorporation of communication networking technologies with legacy industrial control systems have exposed these to outside world. The secure operation of such systems requires novel security solutions as the threat models are different from cyber only systems [16]. Developing new theory to detect these and other attacks has been the focus of research in computer science, systems and control engineering, and other fields [2–5, 7, 9–11, 13, 14].

In this manuscript, we look into security threats in a water treatment testbed. These plants are spread over vast geographical areas, where the physical process is controlled based on remote sensor readings received over the communication networks. However, an attacker might change those sensor measurements which could lead to an undesired control. Several attacks have been reported on water systems in ICS-CERT report [6]. Most of the control theoretic approaches on secure CPS are based on the dynamic system model of the physical process.

This work was supported in part by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-040) and administered by the National Cybersecurity R&D Directorate.

A residual signal is obtained by subtracting the sensor measurements from the sensor estimates (obtained using the system model). An anomaly is detected based on the statistical properties of this residual signal. A large proportion of the literature considers attacks that are executed and attempted to be detected on the same portion of the system, however, many CPS systems are large-scale multistage processes in which the whole process is subdivided into several inter-connected stages. Typically each stage is dependent on the previous stage of the plant, thus it is interesting to model systems at the multistage level [1, 7, 9, 13]. For this case study, we work on a 6 stage water treatment testbed as explained in Sect. 4. In the work presented here, we show the ability to detect attacks that occur in previous stages of the plant, thereby exploiting the coupling between the stages through the physical process. By extensive experimentation on a real testbed, we have shown that due to multistage combined estimation, the detectors on either stage would detect the executed attacks on another different stage. Two major contribution of our work are, (a): Proposed a multistage attack detection scheme, (b): Implementation of the proposed scheme on a real world testbed.

2 Background and System Model

We consider a Linear Time Invariant (LTI) stochastic process of the form:

$$\begin{cases} x(t_{k+1}) = Fx(t_k) + Gu(t_k) + v(t_k), \\ y(t_k) = Cx(t_k) + \eta(t_k), \end{cases} \quad (1)$$

with sampling time-instants t_k , $k \in \mathbb{N}$, state $x \in \mathbb{R}^n$, measured output $y \in \mathbb{R}^m$, control input $u \in \mathbb{R}^1$, matrices F , G , and C of appropriate dimensions, and i.i.d. multivariate zero-mean Gaussian noises $v \in \mathbb{R}^n$ and $\eta \in \mathbb{R}^m$ with covariance matrices $R_1 \in \mathbb{R}^{n \times n}$, $R_1 \geq 0$ and $R_2 \in \mathbb{R}^{m \times m}$, $R_2 \geq 0$, respectively. The initial state $x(t_1)$ is assumed to be a zero-mean Gaussian random vector with covariance matrix $R_0 \in \mathbb{R}^{n \times n}$, $R_0 \geq 0$. The processes $v(t_k)$, $k \in N$ and $\eta(t_k)$, $k \in N$ and the initial condition $x(t_1)$ are mutually independent. At the time-instants t_k , $k \in N$, the output of the process $y(t_k)$ is sampled and transmitted over a communication channel. In this paper, we focus on attacks on sensor measurements by spoofing the signals coming from the sensors to the controller. After each transmission and reception, the attacked output \bar{y} takes the form:

$$\bar{y}(t_k) := y(t_k) + \delta(t_k) = Cx(t_k) + \eta(t_k) + \delta(t_k), \quad (2)$$

where $\delta(t_k) \in \mathbb{R}^m$ denotes additive sensor attacks. Define $x_k := x(t_k)$, $u_k := u(t_k)$, $v_k := v(t_k)$, $y_k := y(t_k)$, $\eta_k := \eta(t_k)$, and $\delta_k := \delta(t_k)$.

Residual-based detection mechanisms require an estimator of the system state; here we use the steady state Kalman Filter:

$$\hat{x}_{k+1} = F\hat{x}_k + Gu_k + L(\bar{y}_k - C\hat{x}_k), \quad (3)$$

with estimated state $\hat{x}_k \in \mathbb{R}^n$, $\hat{x}_1 = E[x(t_1)]$, where $E[\cdot]$ denotes expectation, and gain matrix $L \in \mathbb{R}^{n \times m}$. The estimation error, $e_k := x_k - \hat{x}_k$, is governed by the following difference equation

$$e_{k+1} = (F - LC)e_k + v_k - L\eta_k - L\delta_k. \quad (4)$$

If pair (F, C) is detectable, the observer gain L can be selected such that $(F - LC)$ is Schur. Moreover, under detectability of (F, C) , the covariance matrix $P_k := E[e_k e_k^T]$ converges to steady state (in the absence of attacks) in the sense that $\lim_{k \rightarrow \infty} P_k = P$ exists. For $\delta_k = \mathbf{0}$ and given L (such that $(F - LC)$ is Schur), it can be verified that the asymptotic covariance matrix $P = \lim_{k \rightarrow \infty} P_k$ is given by the solution P of the following Lyapunov equation: $(F - LC)P(F - LC)^T - P + R_1 + LR_2L^T = \mathbf{0}$ where $\mathbf{0}$ denotes the zero matrix of appropriate dimensions. It is assumed that the system has reached steady state before an attack occurs.

The estimator predictions are compared with sensor measurements \bar{y}_k which potentially include attacks. If the difference between what is measured and the estimation is larger than expected, there may be a fault in or attack on the system. Define the residual random sequence r_k , $k \in \mathbb{N}$ as

$$r_k := \bar{y}_k - C\hat{x}_k = Ce_k + \eta_k + \delta_k, \quad (5)$$

For this residual, we formulate a one-sided hypothesis where we either accept or reject the null hypothesis that there are no attacks, in which case, the distribution of the residual is zero mean with the attack-free variance.

2.1 Detection Methods

We consider a dedicated detector on each sensor. Throughout the rest of this paper we will reserve the index i to denote the sensor/detector, $i \in \mathcal{I} := \{1, 2, \dots, m\}$. With C_i being the i -th row of C and $\eta_{k,i}$ and $\delta_{k,i}$ denoting the i -th entries of η_k and δ_k , respectively. We propose the absolute value of the entries of the residual sequence as distance measures:

$$z_{k,i} := |r_{k,i}| = |y_{k,i} - C_i x_{k,i} + \delta_{k,i}| = |C_i e_k + \eta_{k,i} + \delta_{k,i}|. \quad (6)$$

Note that, if there are no attacks, $|r_{k,i}|$ follows a *half-normal distribution* [15].

CUSUM Detector: $S_{k,i} = 0$, $i \in l := \{1, 2, \dots, m\}$,

$$\begin{cases} S_{k,i} = \max(0, S_{k-1,i} + |r_{k,i}| - b_i), & \text{if } S_{k-1,i} \leq \tau_i, \\ S_{k,i} = 0 \text{ and } \bar{k}_i = k - 1, & \text{if } S_{k-1,i} > \tau_i, \end{cases} \quad (7)$$

with bias $b_i \in \mathbb{R}_{>0}$, detection threshold $\tau_i \in \mathbb{R}_{>0}$, and alarm time(s) \bar{k}_i . The idea is that the test sequence $S_{k,i}$ accumulates $|r_{k,i}|$ and alarms are triggered when $S_{k,i}$ exceeds the threshold τ_i . Once the bias is chosen, the threshold τ_i must be selected to fulfill a desired false alarm rate A_i^* [12].

Bad-Data Detector:

$$\text{If } |r_{k,i}| > \alpha_i, \quad \bar{k}_i = k, i \in I. \quad (8)$$

where $\alpha_i \in \mathbb{R}_{>0}$ is the detection threshold and \bar{k}_i are the alarms time(s). In this case, the idea is that alarms are triggered if $|r_{k,i}|$ exceeds the threshold α_i . Similar to the CUSUM procedure, the parameter α_i is selected to satisfy a required false alarm rate A_i^* .

3 Attacker Model

In this section, we introduce the attacks launched on the system. A usual attacker model for CPSs encompasses the *intentions and goals* of the attacker [16]. Attacker's intentions may vary from damaging components to changing a system property or performance degradation. It is assumed that the attacker has access to real-time sensor measurements. It also has perfect knowledge of the system dynamics, the control inputs, and the implemented detection procedures. We launch attacks on the two tanks (Tank-A, Tank-B) subsystem of the SWaT as shown in Fig. 1. We consider a man-in-the-middle (*MitM*) attacker profile [17]. This attacker is able to get access to level sensor readings from Tank-A and Tank-B in real-time and inject signals. Three attacks (corresponding to three different injected signals) are considered and implemented on Tank-A of the real water treatment facility:

Constant Bias Injection Attack: In such an attack, the attacker adds constant offsets to true sensor measurements, i.e., $\delta_{k,i} = \bar{\delta}_i \in \mathbb{R}$. Thus, the controller receives an attacked sensor measurement of the form $\bar{y}_{k,i} = y_{k,i} + \bar{\delta}_i$, where $\bar{\delta}_i$ denotes the false data injected by the attacker to sensor i . As we will see later in results section, the constant bias attack is easily detected using the proposed detection methods.

Zero-Alarm Attack for Bad-Data Detector: This attack is designed to stay undetected by the Bad-Data detectors. Because the attacker knows the system dynamics, has access to sensor readings, and knows the detector parameters, it is able to inject false data into real-time measurements and stay undetected. Consider the Bad-Data procedure and write (8) in terms of the estimated state \hat{x}_k :

$$|r_{k,i}| = |y_{k,i} - C_i \hat{x}_{k,i} + \delta_{k,i}| \leq \alpha_i, \quad i \in \mathcal{I}. \quad (9)$$

By assumption, the attacker has access to $y_{k,i} = C_i y_k + \eta_{k,i}$. Moreover, given its perfect knowledge of the observer, the opponent can compute the estimated output $C_i \hat{x}_k$ and then construct $y_{k,i} - C_i \hat{x}_{k,i}$. It follows that

$$\delta_{k,i} = C_i \hat{x}_{k,i} - y_{k,i} + \alpha_i - \epsilon_i, \quad (\alpha_i > \epsilon_i) \rightarrow |r_{k,i}| = \alpha_i - \epsilon_i, \quad i \in \mathcal{I}, \quad (10)$$

is a feasible attack sequence given the capabilities of the attacker. The constant $\epsilon_i > 0$ is a small positive constant introduced to account for numerical precision.

These attacks maximize the damage to the CPS by immediately saturating and maintaining $|r_{k,i}|$ at the constant $\alpha_i - \epsilon_i$. Therefore, for this attack, the sensor measurements received by the controller take the form:

$$\bar{y}_{k,i} = C_i \hat{x}_{k,i} + \alpha_i - \epsilon_i. \quad (11)$$

Zero-Alarm Attack for CUSUM Detector: This attack is designed to stay undetected by the CUSUM detectors. Consider the CUSUM procedure and write (7) in terms of the estimated state \hat{x}_k :

$$S_{k,i} = \max(0, S_{k-1,i} + |y_i - C_i \hat{x}_k + \delta_{k,i}| - b_i), \quad (12)$$

if $S_{k-1,i} \leq \tau_i$ and $S_{k,i} = 0$ if $S_{k-1,i} > \tau_i$. As with the Bad-Data procedure, we look for attack sequences that immediately saturate and then maintain the CUSUM statistic at $S_{k,i} = \tau_i - \epsilon_i$ where ϵ_i ($\min(\tau_i, b_i) > \epsilon_i > 0$) is a small positive constant introduced to account for numerical precision. Assume that the attack starts at some $k = k^* \geq 1$ and $S_{k^*-1,i} \leq \tau_i$, i.e., the attack does not start immediately after a false alarm. Consider the attack:

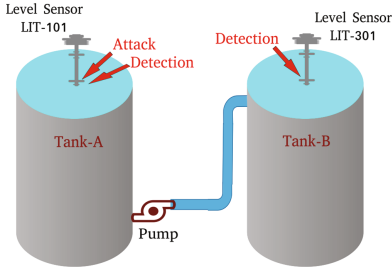
$$\delta_{k,i} = \begin{cases} \tau_i - \epsilon_i + b_i - y_i + C_i \hat{x}_k - S_{k-1,i}, & k = k^*, \\ b_i - y_i + C_i \hat{x}_k, & k > k^*. \end{cases} \quad (13)$$

This attack accomplishes $S_{k,i} = \tau_i - \epsilon_i$ for all $k \geq k^*$ (thus zero alarms). Note that the attacker can only induce this sequence by exactly knowing $S_{k^*-1,i}$, i.e., the value of the CUSUM sequence one step before the attack. This is a strong assumption since it represents a real-time quantity that is not communicated over the communication network. Even if the opponent has access to the parameters of the CUSUM, (b_i, τ_i) , given the stochastic nature of the residuals, the attacker would need to know the complete history of observations (from when the CUSUM was started) to be able to reconstruct $S_{k^*-1,i}$ from data. This is an inherent security advantage in favor of the CUSUM over static detectors like the Bad-Data or Chi-Squared. Nevertheless, for evaluating the worst case scenario, we assume that the attacker has access to $S_{k^*-1,i}$. Therefore, for this attack, the sensor measurements received by the controller take the form:

$$\bar{y}_{k,i} = \begin{cases} C_i \hat{x}_{k,i} + \tau_i - \epsilon_i + b_i - S_{k-1,i} - \epsilon_i, & k = k^*, \\ C_i \hat{x}_{k,i} + b_i, & k > k^*. \end{cases} \quad (14)$$

4 Experimentation Setup

Majority of work on attack detection has considered a single stage for attack and detection (e.g., see [18]). Here, we evaluate the situation of using multiple detectors throughout the process while carrying out a spoofing attack on only one point. In this case we setup the attack on LIT-101 and then we implement a detection mechanism on this tank (LIT-101 at Tank-101) and also on the second



Parameter	Tank-A	Tank-B
α	4.61×10^{-4}	4.59×10^{-4}
τ	1.60×10^{-4}	1.48×10^{-4}
bias b	3.27×10^{-4}	3.26×10^{-4}
\mathcal{A}^*	0.025	0.04

Fig. 1. Two-tank illustration: Tank-101(A), Tank-301(B). The adjoining table reports the parameters for both detectors.

tank (LIT-301 of Tank-301). The challenge in using a process-wide detector is that we require a model that captures not only each stage individually, but also the physical coupling caused by their interconnection. This experiment considers possibly the most obvious of this sort of interconnection and dependency between stages, in the sense that the water out-flow from Tank-101 (Tank-A) should equal the water in-flow to Tank-301 (Tank-B). We can see an illustration of this scenario in Fig. 1. We model the water level and sensor measurements of the two tanks using the following difference equations:

$$\begin{cases} x_{k+1,1} = x_{k,1} + u_{k,1} - u_{k,2}, \\ x_{k+1,2} = x_{k,2} + u_{k,2} - u_{k,3}, \end{cases} \quad \begin{cases} y_{k,1} = x_{k,1} + \eta_{k,1}, \\ y_{k,2} = x_{k,2} + \eta_{k,2}, \end{cases} \quad (15)$$

where $x_{k,j}$, $j = 1, 2$ is the water level at tank j , $u_{k,1}$ and $u_{k,2}$ denote water flowing in and out of tank one, respectively, $u_{k,3}$ is the water flowing out of tank two, and $\eta_{k,j}$ denotes sensor noise. Then, the model of the coupled tanks is of the form (1) with matrices:

$$\left\{ F = R_2 = C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, G = \begin{pmatrix} 1 & -1 & 0 \\ 0 & -1 & 1 \end{pmatrix}, R_1 = R_0 = \mathbf{0}. \right. \quad (16)$$

Having the system model, we can construct a Luenberger observer of the form (3) to estimate the state of the system. The observer matrix L is selected such that the matrix $F - LC$ (with F and C as in (16)) is Schur and its eigenvalues are at 0.5:

$$L = \begin{pmatrix} 0.35 & 0.15 \\ -0.15 & 0.65 \end{pmatrix}.$$

For both detectors, the thresholds (and biases for the CUSUM) have to be selected to satisfy desired false alarm rates \mathcal{A}_j^* . These parameters are selected according to the results in [12] to satisfy a false alarm rate of approximately $\mathcal{A}_1^* = \mathcal{A}_2^* = 0.025$ for both detectors. For our combined detector, we test the obtained detector parameters (shown in Fig. 1) for both the Bad-Data and the CUSUM procedures. We verify by experimenting that for the given parameters, the alarms raised by the detectors converge to $\mathcal{A}^* = 0.04$ (approximately) in the absence of attacks.

5 Performance of Proposed Detectors

We executed the three types of attacks introduced in Sect. 3 with a combined detection procedure running on Tank-A and Tank-B simultaneously.

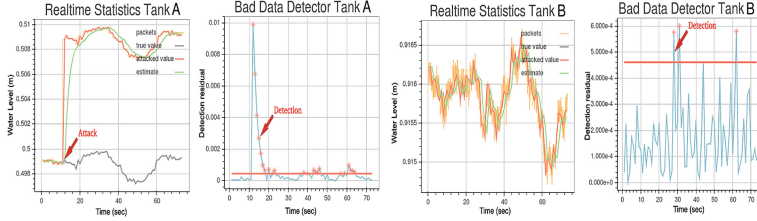


Fig. 2. Constant bias attack detection by combined Bad-Data detector

Constant Bias Attack Detection. Figure 2 shows the water level at the tanks when the system is under a constant bias attack of $\bar{\delta}_1 = 0.01$ m. The PLC received this attacked measurement value with Bad-Data detectors running on both tanks. The true value (plotted in gray) of the level at Tank-A is about 0.5 m. This true level remains constant throughout the attack and the inlet pump and valve are switched OFF. The attack is launched at $k = 11$ s (time instant in plot) and the Bad-Data detector monitoring Tank-A detects it immediately. The Bad-Data detector monitoring Tank-B detects the attack at $k = 28$ s. This proves that the combined detection procedure for Bad-Data detector works well. Furthermore this attack was also detected by the CUSUM detectors running at Tank-A and Tank-B.

Zero-Alarm Attack for Bad-Data Detector. We now test a zero-alarm Bad-Data attack on Tank-A. Both the detector types (i.e., Bad-Data detector and CUSUM detector) monitor Tank-A and Tank-B. In our proposed scheme, when an attack is launched at a single stage, it can be detected by a detector running on another stage. Here we launch a zero-alarm attack against the Bad-Data detector at Tank-A, and found that it can be detected only by CUSUM detector at Tank-A and by both detectors (CUSUM and Bad-Data) running at Tank-B. For The attacker to remain undetected at Tank-A he have to spoof the sensor value according to Sect. 3.

Zero-Alarm Attack for Bad-Data and CUSUM Detector. The last attack type which we executed is the zero-alarm attack for Bad-Data and CUSUM detector. Since this attack is designed to raise no alarms for the Bad-Data or the CUSUM detectors, neither detector on Tank-A detects the attack. The attacker has the complete knowledge of the detectors running on Tank-A, so he can deviate the level of the tank in such a way that Bad-Data detector and CUSUM detector at Tank-A would not be able to detect it, but the combined estimate and detection of the two-tank multistage process makes it possible to detect this attack by the detectors at Tank-B. The result is shown in Fig. 3.

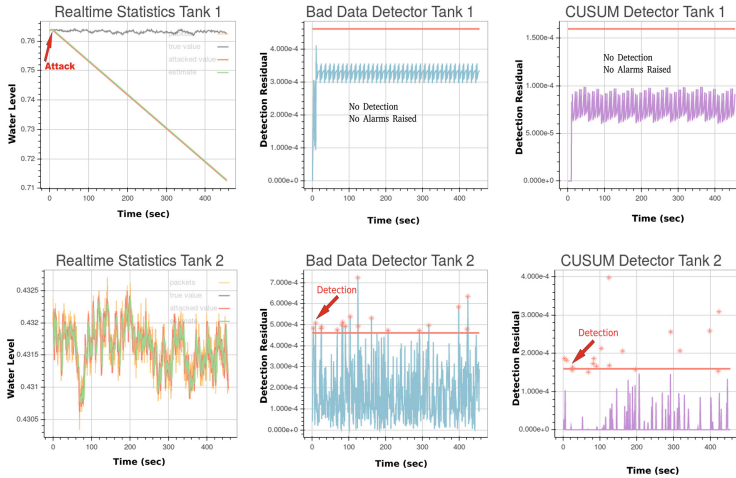


Fig. 3. Zero-alarm Bad-Data/CUSUM detection by combined detectors at Tank-B.

6 Conclusion

In this paper we have provided a real-life experimental case-study about how a multistage detection procedure could be very useful. We showed that proper modeling of the system and the selection of right parameters for detection threshold are very important. Our study points out the limitations of statistical anomaly detectors towards stealthy attacks which are intelligently designed to raise no alarms (zero-alarm attacks). However, we can still use these detection methods if physics of the system is properly integrated in the model for system dynamics. Due to state inter-dependencies, an attacker can hide itself in one stage but its effects can be seen in the following stages. Our results show that it is possible to detect zero-alarm attacks using the proposed scheme.

References

1. Ahmed, C.M., Sridhar, A., Aditya, M.: Limitations of state estimation based cyber attack detection schemes in industrial control systems. In: IEEE Smart City Security and Privacy Workshop, CPSWeek (2016)
2. Ahmed, C.M., Murguia, C., Ruths, J.: Model-based attack detection scheme for smart water distribution networks. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS 2017, pp. 101–113. ACM, New York (2017). <http://doi.acm.org/10.1145/3052973.3053011>
3. Bai, C.Z., Gupta, V.: On Kalman filtering in the presence of a compromised sensor: fundamental performance bounds. In: 2014 American Control Conference, pp. 3029–3034. IEEE (2014)
4. Bai, C.Z., Pasqualetti, F., Gupta, V.: Security in stochastic control systems: fundamental limitations and performance bounds. In: 2015 American Control Conference (ACC), pp. 195–200. IEEE (2015)

5. Cardenas, A.A., Amin, S., Lin, Z.S., Huang, Y.L., Huang, C.Y., Sastry, S.: Attacks against process control systems: risk assessment, detection, and response. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 355–366. ACM (2011)
6. ICS-CERT: Ics-mm201408: May-august 2014. Report no., U.S. Department of Homeland Security-Industrial Control Systems-Cyber Emergency Response Team, Washington, D.C. (2014). <https://ics-cert.us-cert.gov>
7. Kwon, C., Liu, W., Hwang, I.: Security analysis for cyber-physical systems against stealthy deception attacks. In: American Control Conference (ACC), pp. 3344–3349 (2013)
8. Lee, E.A.: Cyber physical systems: design challenges. EECS Department, University of California, Berkeley, Technical report UCB/EECS-2008-8, January 2008. <http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-8.html>
9. Miao, F., Zhu, Q., Pajic, M., Pappas, G.J.: Coding sensor outputs for injection attacks detection. In: IEEE Conference in Decision and Control (CDC), pp. 5776–5781 (2014)
10. Mo, Y., Garone, E., Casavola, A., Sinopoli, B.: False data injection attacks against state estimation in wireless sensor networks. In: 49th IEEE Conference on Decision and Control (CDC), pp. 5967–5972. IEEE (2010)
11. Mujeeb, C.A., Mathur, A.P.: Hardware identification via sensor fingerprinting in a cyber physical system. In: 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 517–524. IEEE (2017)
12. Murguia, C., Ruths, J.: Characterization of a CUSUM model-based sensor attack detector. In: 55th IEEE Conference on Decision and Control Conference (CDC) (2016)
13. Murguia, C., Ruths, J.: CUSUM and chi-squared attack detection of compromised sensors. In: 2016 IEEE Conference on Control Applications (CCA), pp. 474–480, September 2016
14. Pasqualetti, F., Dörfler, F., Bullo, F.: Attack detection and identification in cyber-physical systems. *IEEE Trans. Autom. Control* **58**(11), 2715–2729 (2013)
15. Ross, M.: Introduction to Probability Models, 9th edn. Academic Press Inc., Orlando (2006)
16. Sridhar, A., Aditya, M.: Generalized attacker and attack models for cyber physical systems. In: 40th IEEE COMPSAC (2016)
17. Urbina, D., Giraldo, J., Tippenhauer, N.O., Cardenas, A.: Attacking Fieldbus communications in ICS: applications to the SWaT Testbed. In: Singapore Cyber-Security Conference (SG-CRC), vol. 14, pp. 75–89 (2016)
18. Urbina, D.I., Giraldo, J., Cardenas, A.A., Tippenhauer, N.O., Valente, J., Faisal, M., Ruths, J., Candell, R., Sandberg, H.: Limiting the impact of stealthy attacks on industrial control systems. In: ACM CCS 3(iii) (2016)

Security and Privacy Requirements Assurance and Evaluation

A UML Profile for Privacy-Aware Data Lifecycle Models

Majed Alshammari^(✉) and Andrew Simpson

Department of Computer Science, University of Oxford, Wolfson Building,
Parks Road, Oxford OX1 3QD, UK
{majed.alshammari, andrew.simpson}@cs.ox.ac.uk

Abstract. Concerns over data-processing activities that may lead to privacy violations or harms have motivated the development of legal frameworks and standards to govern the processing of personal data. However, it is widely recognised that there is a disconnect between policy-makers' intentions and software engineering reality. The Abstract Personal Data Lifecycle (APDL) model, which was proposed to serve as an abstract model for personal data life-cycles, distinguishes between the main operations that can be performed on personal data during its life-cycle by outlining the various distinct activities for each operation. We show how the APDL can be represented in terms of the Unified Modeling Language (UML). The profile is illustrated via a realistic case study.

1 Introduction

Privacy concerns have motivated the development of legal frameworks and standards for governing the processing of personal data, such as the EU General Data Protection Regulation (GDPR) [23] and the Global Privacy Standard (GPS) [5]. Typically, legal frameworks and standards are given at a high level of abstraction without relying on rigorous models that explicitly specify privacy-related concepts and associated properties [3]. In parallel, Privacy by Design (PbD) [6] has been advocated as a proactive and integrative approach for embedding privacy into the early stages of the design process. However, the principles of PbD are given at a high level of abstraction, which, in turn, leads to challenges with regards to translating these principles into engineering activities and artefacts [8].

One attempt to address these challenges and bridge the gap between policy-makers and software engineers is a set of activities for privacy engineering [8]. However, these are not accompanied by guidelines that aid engineers in analysing functional requirements or identifying potential privacy risks in a structured and contextual manner. As another example, the privacy design strategies of [9] jump directly from abstract legal principles into software architecture design [13] without providing criteria for making architectural choices or for justifying these choices with respect to a privacy risk analysis [10].

To realise PbD and translate its principles into engineering activities, a systematic methodology for Privacy Impact Assessments (PIAs) is required [16].

A PIA is defined as a process that identifies and mitigates the impact of an initiative on privacy with multiple stakeholders' participation [16, 25]. Accordingly, the first step of a PIA is to describe data-processing activities so that potential privacy risks can be analysed and assessed [16]. Additionally, to support a meaningful participation of multiple stakeholders, a common language needs to be provided to facilitate communication between those stakeholders. As such, the first step towards bridging the gap between policy-makers and software engineers involves providing a data management model built upon an appropriate conceptual model for privacy engineering. Such a model serves as a stepping stone for modelling privacy-related concepts along with associated properties and relationships, and for representing data-processing activities in a way that is amenable to analysis. The Abstract Personal Data Lifecycle (APDL) model [1] was developed to achieve these goals. It represents the processing of personal data in terms of states (data items), operations (data-processing activities), and roles (actors). It facilitates understanding of the meaning of privacy-related concepts and the ways in which systems can be developed to comply with legal frameworks and standards, and to meet data subjects' expectations by supporting the traceability and management of the flow of personal data. The APDL model [1], however, was informally represented in terms of the lifecycle stages, associated activities and involved actors. In order for it to be integrated into an appropriate software engineering process, a widely-used modelling notation needs to be adopted to help support its main concepts. The Unified Modeling Language (UML) [15] is ideal for this purpose.

UMLsec [11] was introduced as a profile to represent security-related concepts using the standard extension mechanisms of the UML meta-model, i.e. stereotypes, tagged values and constraints. Stereotypes are used, along with tagged values to represent the key aspects of security as requirements and assumptions, and constraints associated to stereotypes give criteria for the evaluation of these aspects in a given specification to determine whether the requirements are satisfied [12]. We use the same extension mechanisms to represent the privacy-related concepts introduced by the APDL model. Specifically, we present a conceptual model, upon which we define a profile that allows the APDL model to be represented in the UML as a meta-model. The meta-model represents the personal data lifecycle in terms of stages that involve data-processing activities that consist of concrete actions and corresponding events, and roles that define a set of responsibilities performed by different actors according to their capabilities.

2 The Abstract Personal Data Lifecycle Model

We start by providing an overview of the Abstract Personal Data Lifecycle (APDL) and by stating a set of principles for a conceptual model.

2.1 An Overview of the APDL Model

The APDL model [1] was proposed to serve as an abstract model for personal data lifecycles—where a data lifecycle is defined by a set of stages through

which personal data moves during its lifetime, associated activities, and involved actors. It aims to specify and represent the minimum amount of personal data, along with possible data-processing activities that are necessary for the specified purpose.

The APDL model distinguishes between the main types of operations that can be performed on personal data during its lifetime. For each operation, it outlines distinct activities that can be conducted in an ordered and planned manner. The model categorises these activities into the following lifecycle stages: initiation, collection, retention, access, review, usage, disclosure and destruction. Each stage involves a set of activities that may be performed by one or more types of role: data modellers, data subjects, data controllers, data processors and third parties. Each role specifies a set of related activities that are expected to be performed together by different actors according to their capabilities and responsibilities. Such a classification reflects the extent to which the flow of personal data is appropriate in terms of involved actors and their assigned roles and responsibilities. Each lifecycle stage is restricted by a set of the GPS principles [5] to govern the behaviour of associated activities. For example, ‘purposes’ emphasises that the purposes for which personal data is collected, retained, used and disclosed must be clearly specified and communicated to data subjects at or before the time of collection, and ‘collection limitation’ affirms that the collection of personal data must be fair, lawful and limited to the specified purposes.

2.2 Principles

In order to develop an appropriate conceptual model, essential principles for the core parts of the model, which will be used as the basis for our UML profile, need to be specified. Crucially, the *purpose* and *scope* of modelling shall be specified in relation to the context of data protection. In addition, the *most appropriate technique* shall be used for deriving useful and potentially usable concepts, associated properties and relationships.

We partially specify the purpose and scope of the modelling, as well as the appropriate techniques and conceptualisation approach used.

1. The purpose of building a conceptual model is to describe precisely the key privacy-related concepts, associated properties and relationships in the context of data protection. The model is intended to be used by multiple stakeholders—both those concerned with data protection and those responsible for developing and maintaining privacy-preserving systems. The model is intended to be used as a common language for privacy engineering to consider protection, manageability and traceability of personal data. Such a language is provided with the ability to express stakeholders’ expectations and concerns.
2. The scope of the modelling is identified by a list of concepts (as explained in Sect. 3).
3. Informal text analysis is used with the aim of analysing commonly-used concepts that have been already described in privacy standards instead of

‘starting from scratch’. We have chosen the Global Privacy Standard (GPS) principles [5] and the Generally Accepted Privacy Principles (GAPP) [2] to ensure that our modelling of the privacy-related concepts and their meanings are based upon a widely-used set of terms. These standards are based on internationally known Fair Information Practice Principles (FIPPs) [24].

4. Concept classification is used with the aim of analysing and classifying relevant terms into concepts and processing activities that can be represented in a fine-grained manner as actions. With regards to concepts, we identify and describe useful and potentially usable concepts, associated properties, meanings and possible values. With regards to actions, we identify and describe useful and potentially usable actions and associated constraints that specify conditions to be satisfied before, or to be guaranteed after, the execution of corresponding actions.

3 A Conceptual Model for the APDL

We now define a conceptual model for the APDL.

3.1 Purpose and Personal Data

Purpose represents goals and reasons for which personal data is collected and processed. It has the following properties: *informalDescription*; *actualPurpose* (in a concrete and explicit manner); *isFair* (indicates whether the processing of personal data has justified adverse effects on the concerned data subjects and is consistent with their reasonable expectations); *isLawful* (indicates whether there are legitimate or legal grounds for collecting and processing personal data); *isProportional* (indicates whether the specified purpose is legally and politically assessed in terms of proportionality); and *relevantPrinciple* (specifies the relevant GPS principles that govern the purpose specification in the sense of placing limitations or constraints).

For a purpose to be fulfilled, a minimum amount of personal data needs to be appropriately specified. As such, **PersonalData** represents the minimum necessary amount of data that is sufficiently related to an identified or identifiable individual in support of the specified purpose. It has the following properties: *informalDescription*; *category* (indicates the category of personal data in terms of its sensitivity and the manner in which it is to be processed, and drawn from {SpecialCategory, Unspecified}¹); and *type* (indicates the type of personal data in relation to the source and manner in which it is created, and drawn from {Collected, Acquired, Derived}).

In order to specify the required data that fulfils the specified purpose, a data model needs to be constructed; **DataModelling** represents the relevant objects, associated properties, relationships and constraints for the purpose of specifying the required data. This representation can be used as shared knowledge by

¹ Personal data, by its nature, is considered sensitive data when it is related to special categories, including racial or ethnic origin, etc. [23].

multiple stakeholders for a specific application. It has the following properties: *subjectDomain*; *modellingPurpose*; *modellingScope* (specifies the set of objects to be represented at an appropriate level of abstraction); and *dataModel* (refers to the model that represents the relevant objects, their properties and relationships to be used as shared knowledge in the domain of interest).

3.2 Lifecycle and Associated Stages

DataLifecycle represents the main characteristics of the personal data lifecycle in terms of the openness of the processed data and the centrality of its underlying system. It has the following properties: *informalDescription*; *isOpen*; and *isCentralised*. Each data lifecycle consists of a set of stages. As such, **LifecycleStage** represents the concept of a generic lifecycle stage that models all possible stages through which personal data moves during its lifecycle in more repetitive and circular flows. The LifecycleStage is abstractly represented as a general classifier that can be used as a classification of all possible stages of the lifecycle. It is mainly used as a target of generalisations, which can be specialised into eight specific classifiers according to associated activities. We consider each in turn.

Initiation represents a complete processing plan that can be referred to before and during the processing of personal data. This plan is a prerequisite for establishing and representing a privacy notice. It has the following properties: *informalDescription*; *specifiedPurpose*; *requiredData* (specifies the minimum amount of personal data as a set of relevant, adequate and not excessive personal data items in support of the specified purpose); *dataSource* (specifies the sources from which personal data items are to be collected, derived or acquired, whether these are internal or external sources); *availableChoice* (describes the choices available to data subjects with regards to the collection, usage and disclosure of their personal data); *consentType* (indicates the type of consent that needs to be obtained in relation to the degree of personal data sensitivity and the manner in which is to be processed, with values drawn from {Explicit, Implicit}); *collectionMethod*; *storageMethod*; *retentionTime* (specifies the necessary period for which personal data is retained to fulfil the specified purpose or as required by applicable laws and regulations); *retrievalMechanism* (specifies the means and the manner in which personal data is to be retrieved or consulted, including query languages, command-line, browser, or graphical user interfaces); *disclosureMechanism* (specifies the means and the manner in which personal data is to be disseminated, transmitted or made available); *destructionMethod*; *applicableRegulation* (indicates the applicable laws and regulations); and *relevantPrinciple* (specifies the relevant GPS principles that govern the stage of lifecycle in the sense of placing limitations or constraints on the associated activities).

Collection represents the act of creating personal data values, whether these are directly recorded or collected from data subjects, or have been acquired from external sources. It has the properties *informalDescription*, *createdData*, *dataSource*, *collectionMethod*, *availableChoice*, *consentType*, and *relevantPrinciple*.

Retention represents the organising, structuring or storing of personal data values in repositories or digital storage media for operational, compliance or

operational recovery purposes. It has the properties *informalDescription*, *retainedData*, *activityType* (drawn from {PrimaryStorage, Archiving, Backup}), *activityPurpose* (drawn from {Operational, RegulatoryCompliance, FutureReference, OperationalRecovery}), *retentionTime*, *storageMethod*, and *relevantPrinciple*.

Access represents the act of specifying, and retrieving or consulting personal data values that are stored in repositories or digital storage media. The aim is to make this data accessible and ready for use in relation to the specified purpose. It has the following properties: *informalDescription*; *retrievedData*; *retrievalMechanism*; and *relevantPrinciple*.

Review represents the act of implementing the access right and rectifying personal data values by data subjects to ensure that their data is accurate, complete and up-to-date. It has the following properties: *informalDescription*; *reviewedData*; *activityType* (drawn from {Adaptation, Alteration, Alignment}); *activityPurpose* (drawn from {Update, Correction}); and *relevantPrinciple*.

Disclosure represents the act of disseminating, making available or transmitting personal data for external use by third parties. It has the following properties: *informalDescription*; *disclosedData*; *activityType* (drawn from {InitialProcessing, FurtherProcessing}); *activityPurpose* (drawn from {TheSpecifiedPurposes, HistoricalPurposes, ScientificPurposes, StatisticalPurposes}); *disclosureMechanism*; and *relevantPrinciple*.

Usage represents the act of using, altering, adapting, refining, aligning or combining personal data items. It has the properties *informalDescription*, *usedData*, *activityType* (drawn from {Adaptation, Alteration, Alignment, Combination}), *activityPurpose* (drawn from {Use, Derivation}), and *relevantPrinciple*.

Destruction represents the act of erasing, destroying, redacting or disposing of personal data. It has the following properties: *informalDescription*; *destroyedData*; *destructionMethod*; *competentAuthority* (drawn from {InternalUnit, ExternalDataDestructionService}); and *relevantPrinciple*.

3.3 Stage Activities, Events and Actions

StageActivity represents data-processing activities that constitute the operations performing on personal data in each stage of the lifecycle. It has the following properties: *informalDescription*; *input*; *output*; *preCondition*; and *postCondition*.

StageEvent represents occurrences that may happen at specific points in time that may have consequences for personal data. It has two properties: *informalDescription* and *category* (*implicit* events occur on the change of states or the passage of some interval of time; *explicit* events occur when an operation is directly requested).

StageAction represents single execution steps within an activity. Actions are the fundamental units that describe personal data processing activities in a fine-grained manner. It has the following properties: *informalDescription*; *inputParameter*; *outputParameter*; *localPreCondition*; and *localPostCondition*.

3.4 Lifecycle Roles and Actors

LifecycleRole represents the way in which a concerned actor participates in a set of related activities of the personal data lifecycle. As such, a role represents a set of responsibilities that are logically related to each other, either by their objectives or by the actors that may play the role. It has the following properties: *informalDescription*; *roleType* (drawn from {DataModeller, DataSubject, DataController, DataProcessor, ThirdParty}); and *responsibility*. Lifecycle roles are identified to cover all stages of the data lifecycle. The processing of personal data in various stages identifies actors in these roles.

LifecycleActor represents an external or internal entity that is capable of, and responsible for, performing the activities of the role to which is assigned. It has the following properties: *informalDescription*; *actorNature* (drawn from {HumanActor, SoftwareAgent}); and *responsibility*.

Figure 1 shows the meta-model of the APDL profile with minimal syntax, omitting the attributes of the classes for simplicity and readability.

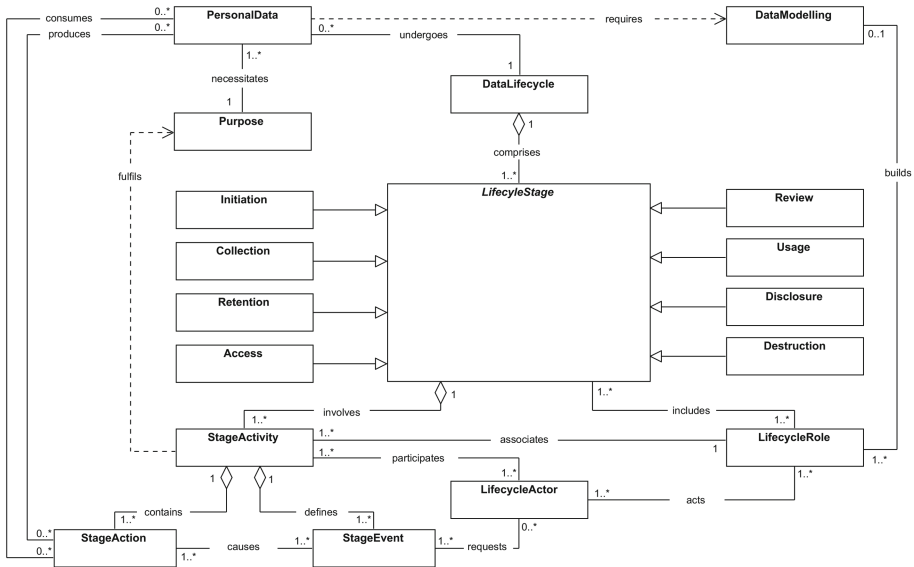


Fig. 1. The meta-model of the APDL profile.

4 A UML Profile for the APDL Model

We now map the conceptual model of Sect. 3 to the UML profile. The stereotypes of the APDL profile are defined to extend existing metaclasses with the aim of using privacy-related terminology whether in place of, or in addition to, the terminology used for the extended metaclasses. The abstract syntax of the APDL is specified by extending three elements of the UML metamodel—the

metaclass *Class*, the metaclass *Association* and the metaclass *Dependency*—with additional properties and constraints. The name of the applied stereotypes are shown within a pair of guillemets. The UML profile defines the concepts needed to model personal data processing activities using UML 2.5 [15]. The constraints needed to express privacy-related concepts of the APDL model are limited to association multiplicities, pre- and post-conditions of stage activities and actions unless additional constraints are explicitly stated.

We distinguish between three levels of abstraction: meta, domain and instance levels. Due to space restrictions, we focus on meta and domain levels only. At the meta-level, we refer to domain-independent abstractions, such as stereotypes, tag definitions, relationships and constraints in relation to the conceptual model. At the domain level, i.e. when a stereotype is applied to a model element, we refer to stereotyped classes, tagged values, domain-specific attributes, relationships and constraints specific to the application domain.

The stereotyped classes and associations belonging to the APDL profile are listed in Table 1.

Table 1. The APDL profile: stereotyped classes and associations.

Ref.	Stereotype	Base class	Parent	Relates
1	«Purpose»	Class	—	—
2	«PersonalData»	Class	—	—
3	«Necessitate»	Association	—	1 → 2
4	«DataModelling»	Class	—	—
5	«Require»	Dependency	—	2 → 4
6	«DataLifecycle»	Class	—	—
7	«Undergo»	Association	—	2 → 6
8	«LifecycleStage»	Class	—	—
9	«Comprise»	Association	—	6 → 8
10	«Initiation»	Class	«LifecycleStage»	—
11	«Collection»	Class	«LifecycleStage»	—
12	«Retention»	Class	«LifecycleStage»	—
13	«Access»	Class	«LifecycleStage»	—
14	«Usage»	Class	«LifecycleStage»	—
15	«Disclosure»	Class	«LifecycleStage»	—
16	«Review»	Class	«LifecycleStage»	—
17	«Destruction»	Class	«LifecycleStage»	—
18	«StageActivity»	Class	—	—

(continued)

Table 1. (continued)

Ref.	Stereotype	Base class	Parent	Relates
19	«Involve»	Association	—	8 → 18
20	«StageEvent»	Class	—	—
21	«Define»	Association	—	18 → 20
22	«StageAction»	Class	—	—
23	«Contain»	Association	—	18 → 22
24	«Fulfil»	Abstraction	—	18 → 1
25	«Produce»	Association	—	22 → 2
26	«Consume»	Association	—	22 → 2
27	«Cause»	Association	—	20 → 22
28	«LifecycleRole»	Class	—	—
29	«Include»	Association	—	8 → 28
30	«Associate»	Association	—	18 → 28
31	«Build»	Association	—	28 → 4
32	«LifecycleActor»	Class	—	—
33	«Act»	Association	—	32 → 28
34	«Participate»	Association	—	32 → 18
35	«Request»	Association	—	32 → 20

4.1 Purpose, PersonalData and DataModelling

Stereotypes. The purpose for which personal data is processed can be specified using the «Purpose» stereotype, which constrains the semantics of the objects, meaning that only they can be used as purposes. The primary tag definitions of the Purpose stereotype are *informalDescription*, *isFair*, *isLawful*, *isProportional* and *relevantPrinciple*, as illustrated in Fig. 2. When the Purpose stereotype is applied to any class, its primary attributes may include *actualPurpose*. Some aspects are more challenging to model, such as the fairness, lawfulness and proportionality of the specified purpose. These can be represented as *Boolean* tagged values to be specified by competent or authorised actors. The primary tag definitions of the «PersonalData» stereotype are *informalDescription*, *category* and

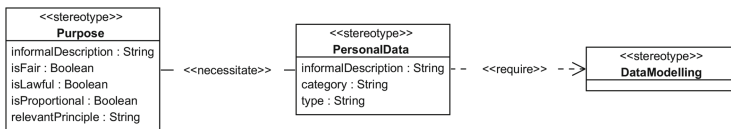


Fig. 2. Purpose, PersonalData and DataModelling stereotypes at the meta-level.

type, as illustrated in Fig. 2. When the `PersonalData` stereotype is applied to any class, its primary attributes are specific to the application domain.

Relationships. «Necessitate» is an association that denotes a relationship between «Purpose» and «PersonalData». It is used to specify a minimum amount of data required to fulfil the specified purpose, as illustrated in Fig. 2. «Require» is a usage dependency that denotes a relationship between «RequiredData» and «DataModelling». It specifies that the required data that fulfils the specified purpose requires a data model for its full specification, as per Fig. 2.

Constraints. Each data-processing initiative has an abstract purpose that can be concretely refined to a set of purposes that can be specified in terms of data-processing activities, which, in turn, can be hierarchically structured as concrete actions and events. Consequently, it should not make sense to have multiple instances of the class stereotyped by «Purpose». As such, it is constrained as a singleton—it is instantiated only once in each particular model.

4.2 DataLifecycle, LifecycleStage and Its Specialisations

Stereotypes. The stages through which personal data moves during its lifetime can be specified using the «DataLifecycle» stereotype. The primary tag definition of the «DataLifecycle» stereotype is: *informalDescription*, *isOpen* and *isCentralised*, as illustrated in Fig. 3. When the `DataLifecycle` stereotype is applied to any class, its primary attributes are specific to the application domain. The `DataLifecycle` stereotype consists of one or more stages that involve data processing activities. These stages are represented by the abstract «LifecycleStage» stereotype. The primary tag definitions of its specialisations are illustrated in Fig. 3.

Relationships. «Undergo» is an association that denotes a relationship between «PersonalData» and «DataLifecycle». The `undergo` association is used to specify that personal data is subject to a set of stages, each of which involves a set of processing activities. «Comprise» is a binary association that denotes a relationship between «DataLifecycle» and «LifecycleStage». The `comprise` association is used to specify that the data lifecycle consists of various stages that involve distinct but related processing activities.

Constraints. «LifecycleStage» represents the concept of a generic lifecycle stage. It is mainly used as a target of generalisation; as such, it is constrained as abstract. The generalisation set, which combines all the special classifiers of the `LifecycleStage`, has two properties: *complete* and *disjoint*. Each data-processing initiative requires the development of a complete processing plan that may serve as the basis of establishing a privacy notice to be communicated to data subjects. Consequently, it should not make sense to have multiple instances of the «Initiation», which represent the processing plan. As such, it is constrained as a singleton.

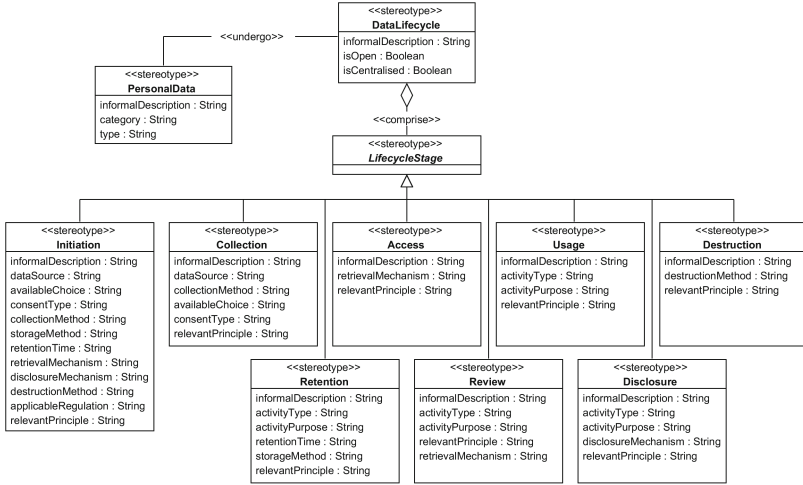


Fig. 3. DataLifecycle and LifecycleStage stereotypes at the meta-level.

4.3 StageActivity, StageEvent and StageAction

Each lifecycle stage involves a set of activities, which, in turn, consist of a set of concrete actions, as well as a set of events that trigger their execution.

Stereotypes. A data-processing activity is specified using the «StageActivity» stereotype, which constrains the semantics of the objects, meaning that only they can be used as processing activities. The primary tag definitions of the Stage-Activity stereotype are *informalDescription*, *preCondition* and *postCondition*, as illustrated in Fig. 4. When the StageActivity stereotype is applied to any class its primary attributes may include *input* and *output*. Each StageActivity contains a set of actions specified by the «StageAction» stereotype. The primary tag definitions of the StageAction stereotype are *informalDescription*, *localPreCondition* and *localPostCondition*, as illustrated in Fig. 4. When the StageAction stereotype is applied to any class, its primary attributes may include *inputParameter* and *outputParameter*. Each StageActivity defines a set of events specified using the «StageEvent» stereotype. The primary tag definitions of the StageEvent stereotype are *informalDescription* and *category*, as illustrated in Fig. 4. When the StageEvent stereotype is applied to any class, its primary attributes are specific to the application domain.

Relationships. Figure 4 shows how LifecycleStage, StageActivity, StageAction and StageEvent stereotypes participate in relationships.

«Involve» is a binary association that denotes a relationship between «LifecycleStage» and «StageActivity». Each lifecycle stage may involve one or more related but distinct activities. «Contain» is a binary association that denotes a relationship between «StageActivity» and «StageAction». Each stage activity may consist of one or more concrete actions. «Define» is a binary association

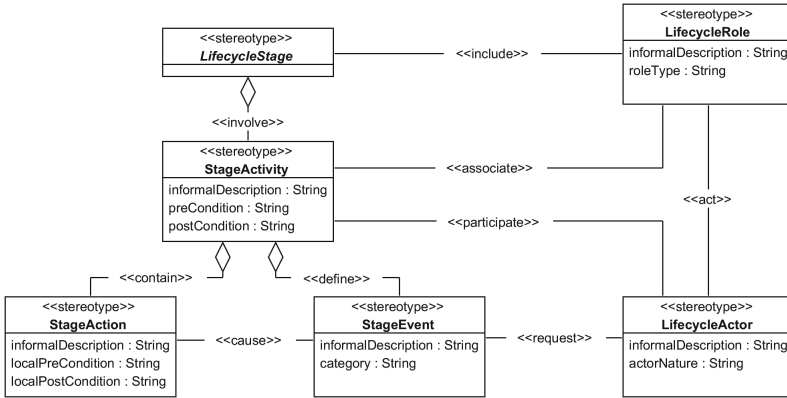


Fig. 4. StageActivity, StageAction, StageEvent, LifecycleRole and LifecycleActor stereotypes at the meta-level.

that denotes a relationship between «StageActivity» and «StageEvent». Each stage activity may consist of one or more events that are requested by actors to trigger its execution. «Cause» is an association that denotes a relationship between «StageEvent» and «StageAction». Each action may be triggered by zero or more events that are requested by actors. «Consume» is an association that denotes a relationship between «StageAction» and «PersonalData». The action requires specific personal data items to accomplish its execution. «Produce» is an association that denotes a relationship between a «StageAction» class and a «PersonalData» class. The action provides specific personal data items as a result of its execution. «Fulfil» is an abstraction that donates a refinement relationship between «StageActivity» and «Purpose». It is a specialisation of the standard abstraction stereotype «Refine». It is used to specify data-processing activities that have already been specified at a certain level of detail as a purpose. It is used to capture how a processing activity participates in the fulfilment of the specified purpose.

Constraints. The constraint of the «Consume» stereotype is that values of the association’s *inputParameter* property must be attributes of the «PersonalData» stereotyped class. Similarly, the constraint of the «Produce» stereotype is that values of the association’s *outputParameter* property must be attributes of the «PersonalData» stereotyped class.

4.4 LifecycleRole and LifecycleActor

Each lifecycle stage involves a set of roles that may be played by different actors.

Stereotypes. A set of related activities that are expected to be performed together can be represented using the «LifecycleRole» stereotype. The primary tag definitions of the LifecycleRole stereotype are *informalDescription* and *roleType*, as illustrated in Fig. 4. When the LifecycleRole stereotype is applied to any

class, its primary attributes are specific to the application domain. An external or internal entity that is capable of, and responsible for, performing a set of activities associated with the role to which is assigned can be specified using the «LifecycleActor» stereotype. The primary tag definitions of the LifecycleActor stereotype are *informalDescription* and *actorNature*, as illustrated in Fig. 4. When the LifecycleActor stereotype is applied to any class, its primary attributes are specific to the application domain.

Relationships. Figure 4 shows how the LifecycleRole and LifecycleActor stereotypes participate in relationships.

«Include» is an association that denotes a relationship between «LifecycleStage» and «LifecycleRole». Each lifecycle stage may include one or more lifecycle roles that participate to accomplish associated activities, and each lifecycle role may participate in one or more lifecycle stages. «Associate» is an association that denotes a relationship between «StageActivity» and «LifecycleRole». Each stage activity may be assigned to exactly one lifecycle role; further, each lifecycle role may involve one or more stage activities. «Act» is an association that denotes a relationship between «LifecycleActor» and «LifecycleRole». Each lifecycle actor may be assigned to one or more lifecycle roles; further, each lifecycle role may involve one or more lifecycle actors. «Request» is an association that denotes a relationship between «LifecycleActor» and «StageEvent». Each lifecycle actor may perform an action by requesting one or more stage events that trigger its execution, and each stage event may be requested by zero or more lifecycle actor. «Participate» is an association that denotes a relationship between «LifecycleActor» and «StageActivity». Each lifecycle actor is capable of performing one or more stage activities that are assigned to one or more lifecycle roles to which the actor is assigned, and each stage activity may involve one or more lifecycle actors. «Build» is an association that denotes a relationship between «LifecycleRole» and «DataModelling». Each data modeller may construct zero or one data models, and each data model may be constructed by one or more data modellers.

5 The European Electronic Toll Service

5.1 Overview

The aim of the European Electronic Toll Service (EETS) [18] is to support interoperability between electronic road toll systems. We have chosen this case study for the following reasons. First, it has been critically analysed with regards to privacy concerns in the literature [4, 8]. Second, EETS is regulated by Directive 2004/52/EC of the European Parliament and of the Council of 29 April 2004 on the interoperability of electronic road toll systems in the Community [21] and the related Commission Decision 2009/750/EC of 6 October 2009 on the definition of the European Electronic Toll Service and its technical elements [22]. Third, the European Commission provides full details about EETS by publishing a guide as a reference manual for all parties concerned by the Directive and the Decision.

The guide illustrates references and procedures to help the implementation of electronic road toll systems interoperability and EETS [18].

EETS complements national electronic road toll systems to ensure their interoperability. It is intended to cover all domains and objects that are subject to toll, such as road networks, specific sections of roads (e.g. a bridge, a tunnel or a ferry connection), or specific areas offering services (e.g. a parking lot or access to a protected area in a city). It enables road users to easily pay road-usage tolls throughout the Member States with a single subscription contract with a service provider [18].

The main parties involved in the EETS are users, service providers and toll chargers. Service providers are legal entities that grant access to EETS to road users [22]. Toll chargers are public or private organisations that are responsible for levying tolls for the circulation of vehicles in an EETS domain [22]. A user is a natural or legal person who subscribes to a service provider in order to get access to EETS, regardless of nationality, country of residence or the Member State in which the vehicle is registered [22]. By signing a contract, a user needs to provide a set of data—user and vehicle classification parameters—specified by a responsible toll charger, as well as to be informed about the processing of their personal data in relation to applicable law and regulations. Accordingly, the service provider provides the user with an On-Board Unit (OBU) to be installed on-board a vehicle to collect, store, and remotely receive and transmit time, distance and location data over time. This data, together with the user's and vehicle's parameters, are specified to declare the toll of circulating a vehicle in a specific toll domain [18].

Prior to explaining the interfaces, we introduce some relevant concepts [22]. *Toll domain* is a road network or a specific section of a road that is subject to toll. *Toll declaration* is a statement sent to a toll charger to confirm the circulation of a vehicle in its a toll domain. *Toll transaction* is an action or a set of actions in which a toll declaration is sent to a toll charger. *Toll context data* is data defined by the responsible toll charger to describe the location of the toll domain, charging policies, and the format of toll declarations. *Domain statement* is a statement that is developed and maintained by a toll charger to establish a set of conditions for a service provider for accessing its toll domains.

Figure 5 illustrates the essential elements of the EETS architecture, together with interfaces via which data is exchanged between these elements [18]. Service providers and toll chargers are required to implement the relevant interfaces in their OBUs and RSE respectively, as well as in their back-office systems.

Interface 1 is used for data exchange between the OBU and the service provider's back-office systems. *Interface 2* is used for data exchange between back-office systems of the service providers and toll chargers. *Interface 3* is used for data exchange between toll chargers' back-office systems and Roadside Equipment (RSE), including toll declaration and enforcement data. *Interface 4* is used for data exchange between a toll charger's RSE and a service provider's OBU.

EETS provision entails personal data processing, which must be carried out in compliance with the EU Directive 95/46/EC [19] and Directive 2002/58/EC [20].

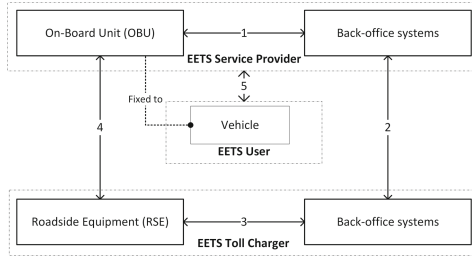


Fig. 5. The essential elements of the EETS architecture.

5.2 An Illustration

We now illustrate the APDL profile by means of our case study.

Purpose, PersonalData and DataModelling. EETSPurpose is a class stereotyped by «Purpose» to represent the main purpose for which EETS users’ personal data is collected and processed. At the domain level, its tagged values informally describe the aim of the stereotyped class, as well as the fairness, lawfulness and proportionality of the purpose, and indicate the relevant GPS principle, as illustrated in Fig. 6. At the instance level, the value of its actualPurpose attribute is to electronically calculate and collect road-usage tolls.

We draw a partial data model diagram that represents the following classes: EETSUser, LocationData, UserAccount, Contract, Vehicle, OBU and TollContextData, as illustrated in Fig. 6. Classes that represent personal data are stereotyped by «PersonalData», whereas those represent generic data are not stereotyped. The relationships in which these classes participate can of course directly be modelled by associations in the UML.

As an example, EETSUser is a class stereotyped by «PersonalData» to represent the user’s personal data, whether the user is the driver, owner, lesser or fleet operator of the vehicle. At the domain level, its tagged values informally describe the aim of the class, the category and the type of the data, as illustrated in Fig. 6. Its primary attributes are userId and billingAddress.

DataLifecycle, LifecycleStage and its specialisations. In reference to the general architecture of the EETS proposed by the European Commission [18], the required personal data needs to be specified by Member States in relation to relevant national regulations. This indicates that the EETS data lifecycle is closed, i.e. no arbitrary data from external sources will be collected, acquired or derived without initial planning. In accordance with the physical nature of the EETS architecture, the collected EETS users’ personal data is not stored in a centrally controlled infrastructure. Thus the EETS data lifecycle is decentralised.

EETSDataLifecycle is a class stereotyped by «DataLifecycle» to represent the main characteristics of the personal data lifecycle in the context of EETS. At the domain level, its tagged values informally describe the aim of the class, the openness of the processed data and the centrality of its underlying system,

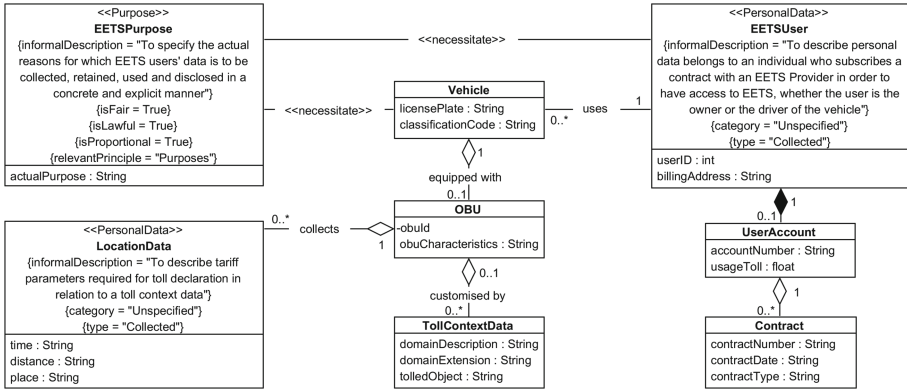


Fig. 6. EETSPurpose, along with the partial data model diagram for EETS.

as illustrated in Fig. 7. At the instance level, the values of its lifecycleType and granularityLevel attributes are Evolutionary and FinedGrained respectively.

Each lifecycle stage can be represented by one or more stereotyped classes that represent sets of activities according to: the types of processed data, its sources, the manner in which it is processed, and the assigned roles and responsibilities of the actors. This gives each lifecycle stage the ability to be expressed in terms of cycles, reflecting the repetitive nature of data processing activities. For example, SubscribingToService and CollectingUsageData are two classes of the collection stage, separated according to the nature of collection activities and responsibilities of the involved actors. To subscribe to an EETS service, drivers, as data subjects, provide their personal data by establishing and signing contracts with the service providers. To collect usage data, EETS service providers, as data processors, collect location data of subscribed EETS users via OBUs.

The SubscribingToService is a class stereotyped by «Collection» to represent a set of related activities, i.e. signing a contract and installing an OBU, with the aim of collecting users’ personal data and vehicles’ classification parameters. At the domain level, its tagged values informally describe the aim of the class, data sources, collection methods, available choices, consent type and the relevant GPS principle, as illustrated in Fig. 7. At the instance level, the values of its createdUserData and createdVehicleData attributes are of the type EETSUser and Vehicle respectively.

The CollectingUsageData is a class stereotyped by «Collection» to represent a set of related activities, i.e. collecting location data, with the aim of collecting the time of usage, the covered distance and the place on which the vehicle is circulating on a particular toll domain for tolls declaration and calculation. At the domain level, its tagged values informally describe the aim of the class, data sources, collection methods, available choices, consent type and the relevant GPS principle, as illustrated in Fig. 7. At the instance level, the value of its createdData attribute is of the type LocationData.

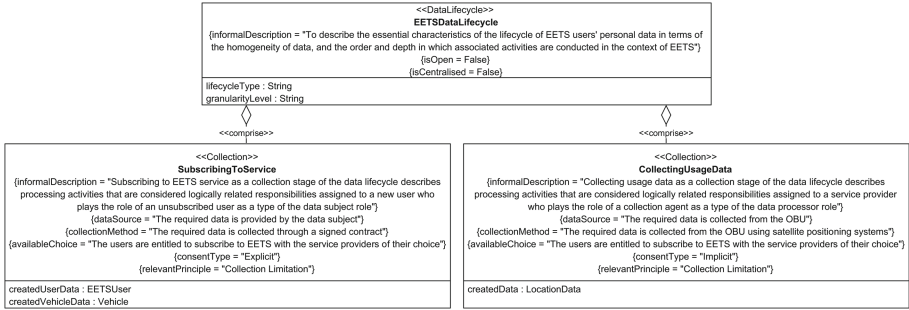


Fig. 7. EETSDataLifecycle and examples of its component stages: SubscribingToService, CollectingUsageData and CheckingOperationalCompliance.

StageActivity, StageEvent and StageAction. Each lifecycle stage involves a set of stage activities, each of which contains a set of actions that represent its executable steps, and a set of events that cause the execution of these actions. SubscribingToService, for example, as a collection stage involves two stage activities: SigningContract and InstallingOBU.

SigningContract is a class stereotyped by «StageActivity» to represent the activity of collecting user’s personal data by signing a contract with a service provider. At the domain level, its tagged values informally describe the aim of the class, the pre- and the post-conditions of the activity, as illustrated in Fig. 8. At the instance level, the values of its input and output attributes are of the type EETSUser. The SigningContract coordinates its execution by containing CollectUserData as an action and defining Sign as an event that causes the execution of this action.

InstallingOBU is a class stereotyped by «StageActivity» to represent the activity of collecting vehicle’s classification parameters by initialising the OBU. At the domain level, its tagged values informally describe the aim of the class, the pre- and post-conditions of the activity, as illustrated in Fig. 8. At the instance level, the values of its input and output attributes are of the type Vehicle. InstallingOBU coordinates its execution by containing CollectVehicleData as an action and defining Subscribe as an event that causes its execution.

LifecycleRole and LifecycleActor. Each lifecycle stage includes a set of lifecycle roles, each of which is played by different actors according to their capabilities and responsibilities. The SubscribingToService, for example, includes UnsubscribedUser as a lifecycle role and Driver as a lifecycle actor.

UnsubscribedUser is a class stereotyped by «LifecycleRole» to represent a type of the data subject role as a set of related activities, i.e. SigningContract and InstallingOBU, that are expected to be performed together for a certain task, i.e. subscribing to EETS service. At the domain level, its tagged values informally describe the aim of the class, and the main role type, as per Fig. 8.

Driver is a class stereotyped by «LifecycleActor» to represent the individual capable of, and responsible for, performing the activities of SigningContract and

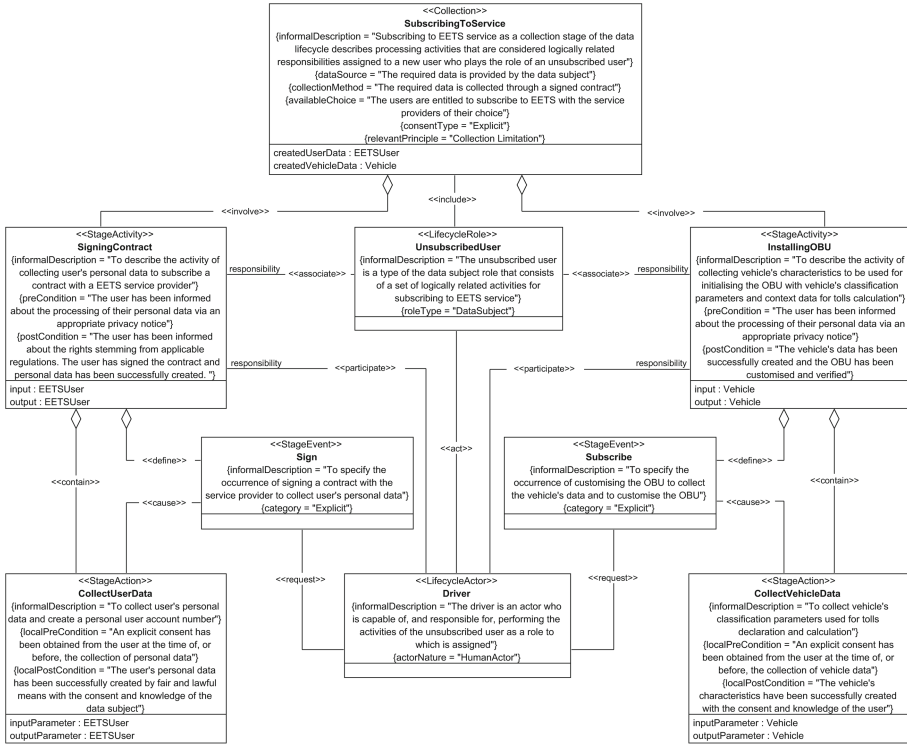


Fig. 8. SubscribingToService as a collection stage, along with its activities, actions, events, roles and actors.

InstallingOBU, i.e. grouped as UnsubscribedUser, with the aim of subscribing to the EETS service. The performance of these activities can be achieved by requesting the Sign and Subscribe events that cause the execution of the CollectUserData and CollectVehicleData actions respectively. At the domain level, its tagged values informally describe the aim of the class and the nature of the actor, as illustrated in Fig. 8.

In summary, the approach has served as a preliminary acquisition step to capture all required concepts that support requirements analysis—a critical step in the system development lifecycle. It represents data-processing activities in a contextual and fined-grained manner that is amenable to risk analysis and compliance checking.

6 Conclusions

The UML profile for the APDL model complements the contributions of [8,9], by providing foundations for analysing functional requirements and assessing potential privacy risks in a contextual and comprehensive manner. It represents

privacy-related concepts using the standard extension mechanisms of the UML meta-model. Stereotypes and tagged values are used to represent key aspects of privacy principles as requirements, and constraints provide criteria for the evaluation of these aspects to determine whether the representation of data-processing activities fulfils these requirements.

The lifecycle stages and roles can be used to classify processing activities into multiple partitions, according to the nature of these activities, the capabilities and responsibilities of involved actors, the organisational units performing these activities or the geographical location at which these activities are performed. The APDL meta-model is a way of describing data-processing at a fined-grained level, with the possibility of expressing how activities are performed, what are their effects in terms of changes of states, when they take place in terms of lifecycle stages, and where they take place in terms of lifecycle roles. This has the potential to support the application of the principle of separation of duties that manages the responsibility assignment by determining who is responsible for which lifecycle stage and what is the level of authority with respect to the decisions and activities performed when data is collaboratively collected and processed by multiple stakeholders in multiple domains. It also facilitates the application of the principle of data minimisation as a foundational step for privacy engineering by specifying the processed data items in each single atomic action within an activity. This, in turn, helps analyse and restrict the processing of personal data to the minimum amount necessary according to the purpose of each concrete action.

We will build upon this work in a number of ways. First, we will use additional case studies to further validate the approach and highlight its usefulness and practical impact in various domains. Possible examples include ePetitions [7]. Second, we will define a data-centric method to complement a PIA by analysing and assessing potential privacy risks in a comprehensive, contextual and non-reductive manner: comprehensive via the adoption of the APDL meta-model as a basis for contextual analysis; contextual via the adoption of Nissenbaum's contextual integrity framework [14]; and non-reductive via the adoption of Solove's taxonomy of privacy [17].

References

1. Alshammari, M., Simpson, A.C.: Personal Data Management for Privacy Engineering: An Abstract Personal Data Lifecycle Model (2017). <https://www.cs.ox.ac.uk/publications/publication10942-abstract.html>
2. American Institute of Certified Public Accountants and Canadian Institute of Chartered Accountants (AICPA/CICA): Generally Accepted Privacy Principles (2009). <https://www.cippguide.org/2010/07/01/generally-accepted-privacy-principles-gapp/>
3. Antignac, T., Scandariato, R., Schneider, G.: A privacy-aware conceptual model for handling personal data. In: Margaria, T., Steffen, B. (eds.) ISO/IEC JTC1/SC29/WG2 N15400:2016. LNCS, vol. 9952, pp. 942–957. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47166-2_65

4. Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwheide, I., Geuens, C.: PrETP: privacy-preserving electronic toll pricing. In: Proceedings of the 19th USENIX Security Symposium, pp. 63–78 (2010)
5. Cavoukian, A.: Creation of a Global Privacy Standard (2006). <https://www.ipc.on.ca/images/Resources/gps.pdf>
6. Cavoukian, A.: Privacy by design... take the challenge. Office of the Information and Privacy Commissioner of Ontario (2009)
7. Diaz, C., Kosta, E., Dekeyser, H., Kohlweiss, M., Nigusse, G.: Privacy preserving electronic petitions. *Identity Inf. Soc.* **1**(1), 203–219 (2008)
8. Gürses, S., Troncoso, C., Diaz, C.: Engineering privacy by design. *Comput. Priv. Data Prot.* **14**(3), 25 (2011)
9. Hoepman, J.-H.: Privacy design strategies. In: Cuppens-Boualahia, N., Cuppens, F., Jajodia, S., Abou El Kalam, A., Sans, T. (eds.) SEC 2014. IAICT, vol. 428, pp. 446–459. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55415-5_38
10. Joyee De, S., Le Métayer, D.: A refinement approach for the reuse of privacy risk analysis results. In: Proceedings of the Fifth ENISA Annual Privacy Forum (APF 2017), pp. 73–109. Österreichische Computer Gesellschaft (2017)
11. Jürjens, J.: UMLsec: extending uml for secure systems development. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 412–425. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45800-X_32
12. Jürjens, J.: Secure Systems Development with UML. Springer, Heidelberg (2005). <https://doi.org/10.1007/b137706>
13. Martín, Y.S., Del Alamo, J.M., Yelmo, J.C.: Engineering privacy requirements valuable lessons from another realm. In: 2014 IEEE 1st Workshop on Evolving Security and Privacy Requirements Engineering (ESPREE), pp. 19–24. IEEE (2014)
14. Nissenbaum, H.F.: Privacy in Context: Technology, Policy, and the Integrity of Social Life. Stanford University Press, Stanford (2009)
15. Object Management Group: OMG Unified Modeling Language (OMG UML) (2015). <http://www.omg.org/spec/UML/>
16. Oetzel, M.C., Spiekermann, S.: A systematic methodology for privacy impact assessments: a design science approach. *Eur. J. Inf. Syst.* **23**(2), 126–150 (2014)
17. Solove, D.J.: A taxonomy of privacy. *Univ. Pa. Law Rev.* **154**(3), 477–564 (2006)
18. The European Commission: The European Electronic Toll Service (EETS): 2011 Guide for the Application of the Directive on the Interoperability of Electronic Road Toll Systems (2011). http://ec.europa.eu/transport/themes/its/road/application_areas/electronic_pricing_and_payment_en
19. The European Union: Official Journal of the European Communities: Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data (1995). <http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:31995L0046&from=EN>
20. The European Union: Official Journal of the European Communities: Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (2002). <http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32002L0058&from=EN>

21. The European Union: Official Journal of the European Communities: Directive 2004/52/EC Of the European Parliament and of the Council of 29 April 2004 on the interoperability of electronic road toll systems in the Community (2004). <http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32004L0052R%2801%29&from=EN>
22. The European Union: Official Journal of the European Communities: Commission Decision 2009/750/EC of 6 October 2009 on the definition of the European Electronic Toll Service and its technical elements (2009). <http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32009D0750>
23. The European Union: Official Journal of the European Union: General Data Protection Regulation (2016). <http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679>
24. United States Department of Health, Education and Welfare: Secretary's Advisory Committee on Automated Personal Data Systems: Records, Computers and the Rights of Citizens: Report. MIT Press, Cambridge (1973)
25. Wright, D.: The state of the art in privacy impact assessment. *Comput. Law Secur. Rev.* **28**(1), 54–61 (2012)

Evaluation of a Security and Privacy Requirements Methodology Using the Physics of Notation

Vasiliki Diamantopoulou^(✉), Michalis Pavlidis, and Haralambos Mouratidis

School of Computing, Engineering and Mathematics, University of Brighton,
Brighton, UK

{v.diamantopoulou,m.pavlidis,h.mouratidis}@brighton.ac.uk

Abstract. Security and Privacy Requirements Methodologies are considered an important part of the development process of systems, especially for the ones that contain and process a large amount of critical information and inevitably needs to remain secure and thus, ensuring privacy. These methodologies provide techniques, methods, and norms for tackling security and privacy issues in Information Systems. In this process, the utilisation of effective, clear and understandable modelling languages with sufficient notation is of utmost importance, since the produced models are used not only among IT experts or among security specialists, but also for communication among various stakeholders, in business environments or among novices in an academic environment. This paper evaluates the effectiveness of a Security and Privacy Requirements Engineering methodology, namely Secure Tropos on the nine principles of the Theory of Notation. Our qualitative analysis revealed a partial satisfaction of these principles.

Keywords: Security requirements engineering
Privacy requirements engineering · Physics of Notation · Evaluation

1 Introduction

The main objective of security and privacy requirements engineering methodologies is to provide techniques, methods and norms for dealing with each task, during the early stages of the Information Systems (IS) development cycle. Security and privacy requirements engineering methodologies supply researchers with existing information about security and privacy requirements in a thorough manner, providing the necessary context to operate [24]. Thus, it is imperative that security and privacy requirements should be specified at the early stages of an IS development process, since by conducting this at an early stage, the building of such requirements is more efficient and also brings about more robust designs [20].

Visual notations, which are considered as a main element of each methodology, are used in all stages of the Software Engineering (SE) process [25], from requirements engineering through to maintenance. They play a particularly critical role in communicating with end users and customers as they are believed to convey information more effectively to non-technical people than text [2], facilitating human communication and problem solving [15]. Visual representations are based on the exploitation of the capabilities of the human visual system. Diagrams can convey information more concisely [8] and precisely than ordinary language [4, 23]. Information presented visually is also more likely to be remembered due to the picture superiority effect [6, 10].

Despite the major contribution that visual syntax has on the understanding of each methodology, it has been argued that the researchers have ignored or undervalued its role. However, there are findings from various empirical studies which confirm the significant role of the visual form of notations and their positive affection to the comprehension of such methodologies, especially by novices [17, 18, 29, 32]. In this direction, it has been reported [11, 22] that more effort is spent on designing semantics of the methodologies, i.e. what concepts to include and what they mean, while visual syntax, i.e. how to visually represent these concepts, is often considered at a later stage. Notations are usually evaluated based mainly on their semantics, not paying the necessitated attention to visual syntax [30, 35].

Design rationale is the process of documenting design decisions made and the reasons they were made. This provides traceability in the design process and helps justify the final design [17]. Such rationale is conspicuously absent in design of methodologies visual notations. The graphical conventions that have been chosen are typically defined without any reference to respective theory or empirical evidence, or any other justification. However, the definition of explicit principles that transform [25] visual notation design from an *unselfconscious* process into a *self-conscious* process is imperative.

The diagram notation which is used during modelling has received little or no attention, and is regarded to be of secondary importance, probably a matter of taste rather than of science. This could be explained by the fact that researchers consider visual notations as being informal, and that therefore they analyse them only from the perspective of their semantics. However, this can be considered as a misunderstanding, since visual languages are no less formal than textual ones [4, 16]. Also, methods used for analysing visual representations are less mature than those for analysing verbal or mathematical representations [14, 37]. Finally, a third explanation could be that researchers and notation designers consider visual syntax to be insignificant, i.e. decisions about semantics (*content*) are paid high attention, while decisions about visual representation (*form*) are often considered to be a matter of aesthetics rather than effectiveness [17].

Taking all the above into consideration, we evaluate an already existing security and privacy requirements engineering methodology, namely Secure Tropos, regarding the visual notation that is being used. The aim of this study

is to examine the graphical notation of Secure Tropos modelling language in order to further improve it at a later stage. The remainder of the paper is set out as follows: Sect. 2 presents a security and privacy requirements engineering methodology, namely Secure Tropos, focusing on the visual notation that is being used. Section 3 discusses related work while Sect. 4 provides the visual notation guidelines, as they have been defined by the relevant literature. Section 5 evaluates the aforementioned methodology, using the Physics of Notation and finally, Sect. 6 concludes the paper, by raising issues for improvement of the examined methodology.

2 Secure Tropos Methodology

Secure Tropos methodology [27] provides a structured approach for goal-oriented security and privacy requirements, applicable to software systems, either to traditional ones or to cloud computing environments [28]. It is based on social hierarchies and adapts components of the i^* framework [38]. This methodology is intended to support all the analysis and design activities in the software development process, supporting the fully capturing, analysis and reasoning of security and privacy requirements from the early stages of the development process. More specifically, it provides a modelling language that represents security and privacy requirements through constraints, allowing developers to model multi-agent, software systems and their organisational environment. The methodology combines concepts from requirements engineering for representing general concepts, and security and privacy engineering for representing security- and privacy-oriented concepts.

The Secure Tropos methodology closely follows the software development life-cycle, i.e. capturing of early requirements, late requirements, architectural design, detailed design, and finally, implementation. Thus, it allows the developer to create and refine models, starting from the system-as-it-is, in order to finally develop the system-to-be, during the analysis and design stage.

2.1 Secure Tropos Model Views

The Secure Tropos modelling language is based on the concepts that have been defined in the requirements engineering discipline, combined with concepts from the security and privacy requirements engineering, all of whom are presented in Tables 1, 2, 3 and 4. The Secure Tropos produces models that contain security and privacy requirements analysis, but with the support of the corresponding tool, namely SecTro [31], the information is grouped according to three perspectives (views), (i) the Organisational view, (ii) the Requirements view and (iii) the Attacks view. Each view provides specific focus of the system under analysis.

Table 1. Concept types on Secure Tropos methodology - organisational and security requirements view

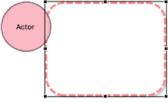




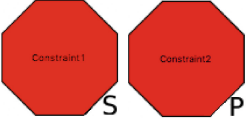
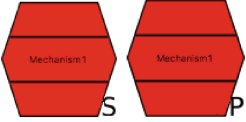




Concept	Description	Notation
Actor	Active entities that carry out actions to achieve goals by exercising its know-how. The term actor refers generically to any unit to which intentional dependencies can be ascribed. An actor interacts with other actors not only through actions or information flows but also relate to each other at an intentional level. Actors depend on each other to achieve goals, perform tasks, and furnish resources. While each actor has strategic goals to pursue, they are achieved through a network of intentional dependencies	
(Hard) Goal	A condition or state of affairs to be achieved. An actor can choose freely 'among different ways to achieve a goal. Represents and intentional desire of an actor, the specifics of <i>how</i> the goal is to be satisfied is not described by the goal. This can be described through task decomposition.	
Soft Goal	A goal that does not have clear criteria on whether it has been achieved.	
Plan	The way to achieve a goal.	
Resource	An informational or physical entity.	
Constraint	A restriction on an actor's function. There are two types of Constraints, namely Security and Privacy. Additionally, a Constraint is related to an Objective e.g., Confidentiality, Integrity, Authentication, etc.	
Mechanism	Represents a system mechanism that supports the satisfaction of a security objective. It can be any of two types, Security or Privacy.	
Threat	Represents a circumstance that has the potential to cause damage to the system.	

Table 2. Concept types on Secure Tropos methodology - security attacks view

Concept	Description	Notation
Attacker	A malicious actor who tries to endanger the security of the system through attacking its resources, goals and plans.	
Vulnerability	A weakness of the system or the organisation.	
Attacks method	A method by which a Threat is realised.	

Organisational View. This view represents the organisational architecture allowing a developer to understand the requirements of the organisation and any interactions between the organisation and external actors or systems. In addition, it displays the organisations’ boundaries, where organisational actors reside; any external actors are modelled outside of this boundary. Organisational view represents the system-as-it-is (Fig. 1).

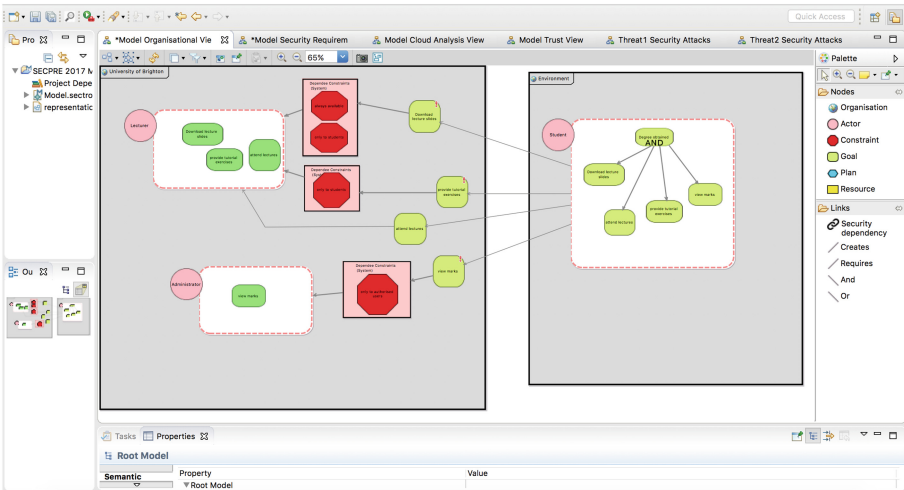


Fig. 1. Organisational view

Table 3. Relationship types on Secure Tropos methodology - organisational and security requirements view


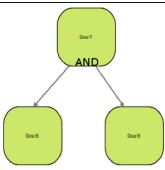
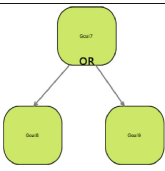
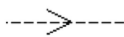

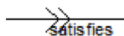
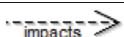
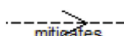
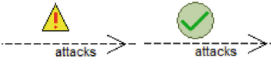
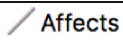
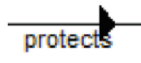
Relation class	Description	Notation
Dependency	The depender depends on the dependee to bring about a certain state of affairs in the world. The dependum is expressed as an assertion statement. The dependee is free to and is expected to make whatever decisions are necessary to achieve the goal (namely, the dependum). The depender does not care how the dependee goes about achieving the goal.	
And	Allows the decomposition of an element to more fine grained elements. All the sub-elements need to be fulfilled in order the parent element to be fulfilled as well. The elements that can be decomposed are a goal, a plan, a resource, a mechanism, an attack method.	
Or	Allows the decomposition of an element to more fine grained elements. The difference with the 'And' relationship is that only one element is needed for the fulfilment of the parent element.	
Contribution	Shows a contribution toward satisfying a soft goal, typically from a task or another soft goal. Any of these Contribution links can be used to link any of the elements to a soft goal to model the way any of these Elements contributes to the satisfaction or fulfilment of the soft goal.	
Restricts	Shows the goal that is restricted by a Security Constraint.	
Satisfies	Shows the Security Constraint that a mechanism satisfies.	
Impacts	Shows the Goal that is affected by a Threat	
Mitigates	Shows the Threat that is mitigated by a Security Mechanism.	

Table 4. Relationship types on Secure Tropos methodology - security attacks view

Relation class	Description	Notation
Attacks	Shows the Vulnerability that an Attack Method is exploiting.	
Affects	Shows what goals and/or resources a vulnerability puts at risk.	
Protects	Shows what mechanisms work as countermeasure.	

Requirements View. This view provides a detailed representation of the organisational view. There, system actors and their goals are designed including the security and privacy analysis concepts. The modelling activity focuses on the responsibilities of the system and other actors, as well as the interaction of actors with the system itself. Requirements view represents the system-to-be (Fig. 2).

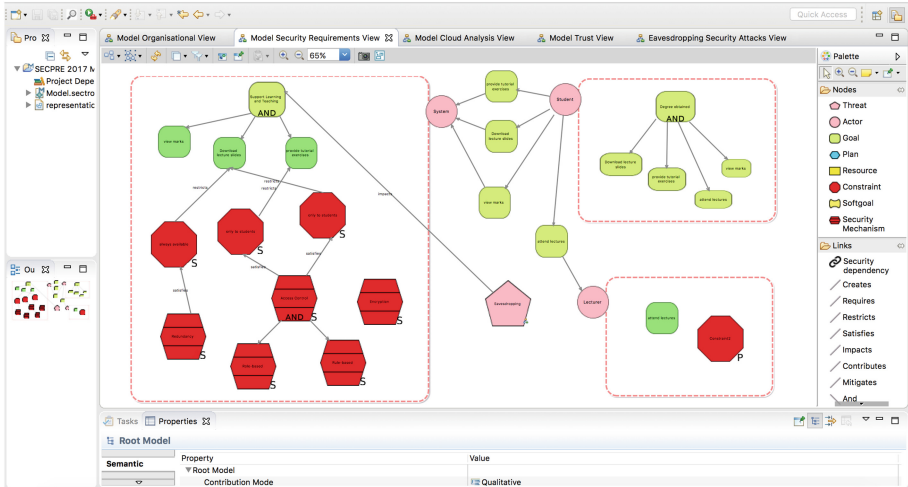


Fig. 2. Requirements view

Attacks View. This view allows the evaluation of the system security and privacy against various attacks. The attack modelling takes place by analysing and checking whether security and privacy threats, which have already been introduced in the Requirements View, are mitigated by the security mechanisms

and privacy enhancing technologies, respectively, available within the system. If the developer identifies any inability of the system to mitigate these threats, they follow an iterative process, going back to the Requirements View, and adjust the design accordingly (Fig. 3).

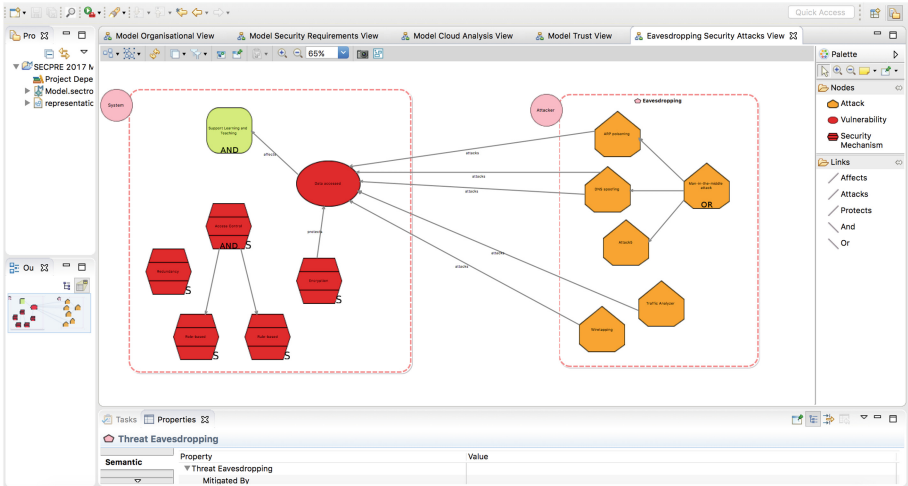


Fig. 3. Attacks view

3 Related Work

In the IT field, one theory of visual notation design that the literature review revealed is the *Cognitive Dimensions Framework* [5, 12, 13]. This framework sets out a vocabulary of terms designed to capture the cognitively-relevant aspects of structure, and shows how they can be traded off against each other, being applied to visual programming environments. Nevertheless, this framework lacks to define theoretical and empirical foundations, it excludes visual representations from its analysis, it does not support evaluation of the chosen notations under evaluation.

Ontological analysis is also accepted for the evaluation of Software Engineering notations [9, 34]. This analysis is conducted through a two-way mapping between a modelling notation and an ontology. Ontological analysis supports the evaluation of the semantics of notations but specifically excludes visual representation aspects, since it focuses on content rather than on form.

The Physics of Notation [7, 25] defines a set of principles for designing cognitively effective visual notations, focusing on the physical properties of notations rather than their logical properties and it is considered as the most prominent and well accepted in the evaluation of software engineering.

4 Visual Notation Principles According to the Theory of Notation

For the effective approach of the evaluation of the graphics of notation, the reader should be aware of specific definitions. A **visual notation** (or visual language, graphical notation, diagramming notation) consists of a set of **graphical symbols (visual vocabulary)**, a set of **compositional rules (visual grammar)** and definitions of the meanings of each symbol (**visual semantics**). The visual vocabulary and visual grammar together form the **visual (or concrete) syntax**. Graphical symbols are used to **symbolise** (perceptually represent) **semantic content**, typically defined by a **metamodel**. The meanings of graphical symbols are defined by mapping them to constructs they represent [25]. A valid expression in a visual notation is called a **visual sentence or diagram**. Diagrams are composed of **symbol instances**, arranged according to the rules of the visual grammar. What has to be addressed in visual notation design is the clear **design goal**. Goals such as simplicity, aesthetics, expressiveness, and naturalness are often mentioned in the literature. In addition, to be most effective in facilitating human communication and problem solving, visual notations need to be optimised for processing by the human mind. Thus, **cognitive effectiveness** is defined as the speed, ease, and accuracy with which a representation can be processed by the human mind [23]. Cognitive effectiveness determines the ability of visual notations to both communicate with business stakeholders and support design and problem solving by software engineers.

According to [25], there are nine principles for designing cognitively effective visual notation. For the development of these principles, information from theory and empirical evidence about cognitive effectiveness of visual representations has been synthesised. More specifically, the nine principles that will be the guide for the evaluation of security requirements methodology are the following:

1. **Principle of Semiotic Clarity:** This principle mentions that there should be an one-to-one correspondence between semantic constructs and graphical symbols. The notations aim at precision, expressiveness, and parsimony, in order for users to effectively design the examined systems.
2. **Principle of Perceptual Discriminability:** This principle mentions that different symbols should be clearly distinguishable from each other. The concepts should have been represented with accurate graphical symbols, easily distinguishable. Consequently, this can lead to the accurate interpretation of the model as a whole [37]. This principle is determined by (i) the *visual distance* between the symbols, i.e. the different visual variables that have been used for the representation of each concept, (ii) the *primacy of shapes*, which contributes to the identification of the objects within a diagram, (iii) the *redundant coding* which contributes to the elimination of errors, (iv) the *perceptual popout* which suggests a unique value on at least one visual variable, and (v) the *textual differentiation*, when the discrimination among the concepts is basically achieved with the use of text and typographic characteristics (fond styles such as bold, italics and underlining).

3. **Principle of Semantic Transparency:** This principle highlights the utilisation of visual representations whose appearance suggests their meaning. The notation that is used should be such, that the user can comprehend the content of the symbol only by its appearance, by providing cues to their meaning. This principle aims to minimise the demanded effort for the understanding of the meaning of a concept.
4. **Principle of Complexity Management:** This principle focuses on diagrams' notation, mentioning that explicit mechanisms for dealing with complexity should be included. The complexity level of a diagram plays an important role in its comprehension, especially when dealing with novices. Excessive complexity is considered a barrier for users to understand SE diagrams [26, 33]. Modularisation and hierarchy are mechanisms that can be used in order to manage complexity in SE notations. More specifically, modularising SE diagrams could result to the improvement of end-users' comprehension. This can be achieved through certain semantic constructs, i.e. subsystem constructs or decomposable constructs. Also, diagrammatic conventions for the decomposition of diagrams should be defined. Regarding hierarchy, it allows systems to be represented at different levels of abstraction and detail, allowing thus, developers to control the complexion at each level.
5. **Principle of Cognitive Integration:** This principle mentions the inclusion of explicit mechanisms to support integration of information from different diagrams. The representation of a system through multiple diagrams demands additional effort by the end user to integrate information from different sources (diagrams). This state has been addressed through (i) conceptual and (ii) perceptual integration. Conceptual integration refers to mechanisms that support the assembling of information from different diagrams into contiguous system representation. Perceptual integration aims to provide the navigation and transition from the one diagram to the other in a simpler and easy for the reader to follow way.
6. **Principle of Visual Expressiveness:** This principle suggests the full range and capacities of visual variables. More specifically, this principle measures visual variation across the entire visual vocabulary [4]. The expression of each concept with the use of a range of visual variables results in the enrichment of the representation that exploits multiple visual communication channels. This principle, which is also related with the one of Perceptual Discriminability, can contribute to the improvement of models understandability. The choice of visual variables should be based on the nature of information that needs to be conveyed [4].
7. **Principle of Dual Coding:** In continuation to both the visual expressiveness and complexity management, this principle suggest the use of text in the modelling process, when the text is used supplementary, rather than as a substitute, i.e. as a form of *redundant coding to reinforce and clarify meaning*. This principle is also based in the differentiated characteristics that humans have regarding their ability to comprehend a meaning. The use of dual coding aims at capturing the human abilities across their full spectrum of spatial and verbal abilities [36].

8. **Principle of Graphic Economy:** This principle refers to the careful number of different graphical symbols that should be used in a methodology. It is argued [21] that the cognitive limits on the number of visual categories that the human mind can effectively recognise are limited. Consequently, the reasonable use of visual categories is proposed, otherwise the users' understandability is negatively affected.
9. **Principle of Cognitive Fit:** This principle highlights the use of different visual dialects for the representation of information, either in case that we deal with different audiences, or in case that we have different representational medium. In the first case, the representation should cover both the expert users and the novices, since they have different level of understandability. In this direction, the approach of the 'lowest common denominator', by using notations understandable by both two types of audience should be avoided, since it can negatively affect the effectiveness for both of the types of users [19]. Regarding the representational medium, this also can affect the communication of the model with the user. More specifically, since there is the option of the representation of a model without the assistance of a CASE tool, the representation of the concepts should be such, to be able to be transferred in 'a piece of paper'. This aspect of the principle of cognitive fit can explain the absence of techniques such as colour, icons, and 3D shapes.

5 Methodology Evaluation

The Secure Tropos graphical notation has not followed specific justification regarding the design choices of the symbols that are used. These design choices have been asserted, following unself-conscious design culture [1], which it is not based on explicit design principles but on instinct, imitation, and tradition. Nevertheless, we proceed with the evaluation of its graphical notation based on the nine principles of the Theory of Notation [25].

Principle of Semiotic Clarity. The concepts and the relationship elements of Secure Tropos, which are presented in Tables 1, 2, 3 and 4, reveal that there is one-to-one correspondence between symbols and their referent concepts. This correspondence contributes to the precision and the efficient expressiveness of the symbols, avoiding the ambiguity and their misinterpretation by the users. Thus, the principle of Semiotic Clarity is fully satisfied.

Principle of Perceptual Discriminability. Regarding the shapes that have been used in order to represent the various concepts in Secure Tropos, the visual distance between the symbols is substantial enough. The identification of the various objects is achieved through the utilisation of the most of the concepts (see Tables 1 and 2) different shapes and colours. The shapes that have been used for the representation of the communication links (see Tables 3 and 4) consist of lines, but with elements that discriminate them (i.e. arrows, dashed lines).

It is argued [25] that most SE notations use a perceptually limited repertoire of shapes, mostly rectangle variants. In the examined methodology we can identify the use of clearly discriminable shapes that represent different constructs; they all come from different shape families and differences between them can be detected pre-attentively. Furthermore, the variable of colour is also used in the concepts of the methodology, improving discriminability between entities, satisfying the *redundant coding* sub-principle. However, the same colour for more than one concepts is being used and this can cause misunderstandings that might incommode the perceptual processing of the user. In addition, Secure Tropos uses text (labels) to differentiate between most of the relationship types. Textual differentiation of symbols is a common but cognitively ineffective way of dealing with excessive graphic complexity, as text processing relies on less efficient cognitive process. Textual differentiation of symbols also confounds the role of text in diagrams. Labels play a critical role at the sentence level in distinguishing between symbol instances and defining their correspondence in the real world. Also, when labels are used to distinguish between relationship types, it precludes the use of user-defined and domain-relevant names. Text is an effective way to distinguish between symbol instances but not between symbol types. Thus, the principle of Perceptual Discriminability is partially satisfied.

Principle of Semantic Transparency. Among all the graphical notations of Secure Tropos, there is one, the “Constraint” which is depicted as a “Stop” sign and satisfies this principle. Stop sign is a familiar signal which can be interpreted as the criticality of a situation. In the same way, the concept of constraints represents a set of restrictions that do not permit specific actions to be taken. Attack link also satisfies this principle, since its depiction is accompanied by two symbols, i.e. a red exclamation mark and a green tick. The first symbol aims to gain user’s attention since an identified vulnerability has not been mitigated by a security or a privacy mechanism, while the second symbol confirms that all possible attacks have been mitigated. Thus, the principle of Semantic Transparency is partially satisfied.

Principle of Complexity Management. In the diagrams of Secure Tropos the design can follow *hierarchy* structure for the representation of goals, in order for the model to be well-structured, and thus contributing to the readability of each model. Moreover, the concept of *modularisation* finds application in Secure Tropos, since, as we described in Sect. 2, there are different views, i.e. Organisational, Requirements, and Attacks, where the information is grouped according to these three perspectives. Through this approach, each model is presented to the user from different viewpoints, improving their understanding. In addition, each identified threat is presented in an additional view, so as the created models are more readable. Thus, the principle of Complexity Management is partially satisfied. However, in a case where the created model is too overloaded with information, the principle of Complexity Management is not satisfied; a problem that is encountered in goal oriented diagrams.

Principle of Cognitive Integration. As we described in the previous principle, in Secure Tropos multiple diagrams are used to represent one system. Each view is responsible for specific analysis of the system-as-it-is and also the system-to-be. Consequently, an end-user needs to parse all the information that has been recorded in each view, in order to have a holistic knowledge of the examined system. Despite this complexity, the notation that the methodology uses is presented in this way that contributes to the elimination of the effort that is demanded by the reader in order to keep track of where they are. The transition from one view to the other can be achieved more smoothly and can constitute to the connection point between different views. Separated tabs support user orientation by indicating where they are in the system of diagrams, allowing easy navigation. Moreover, the concepts that are introduced in the Organisational view (the first view) and are essential for the further analysis to the next two views, are automatically introduced. This results to the facilitation of the user to realise the core concepts of the analysed system. Thus, the principle of Cognitive Integration is fully satisfied.

Principle of Visual Expressiveness. Secure Tropos uses colours in order to distinguish each concept. Colour is not the only identifiable characteristic of each concept, shape is another one. They together facilitate comprehension of the models, avoiding misunderstandings, technical or human related (e.g., black-and-white printing, colour blindness, respectively). In addition, Secure Tropos uses a variety of shapes, i.e. rectangle, rounded rectangle, cycle, hexagon, heptagon, octagon, diamond shape, and ellipse. The literature refers that this variety of shapes is the less effective one regarding human visual processing, and thus curved, 3D, and iconic shapes have to be preferred [3,37]. Regarding the ratio of graphical encoding versus textual encoding, Secure Tropos fails to satisfy this balance (it is argued that the more visual variables that are used, the greater the role of perceptual processing [25]), since textual encoding is used in all of the relationship notations; a point that is not preferred if a model aims to maximise their visual expressiveness. Thus, the principle of Visual Expressiveness is partially satisfied.

Principle of Dual Coding. Each concept of Secure Tropos is depicted both by a graphic symbol and their corresponding label. Moreover, when a concept is inserted to the design space, a Properties panel provides information regarding the specific concept, which can also contribute to the satisfaction of the Dual Coding. In this way, the interpretation of each concept can be achieved with confidence by the user. Thus, the principle of Dual Coding is fully satisfied.

Principle of Graphic Economy. By using the different views of the Secure Tropos, the user is able to focus on a specific perspective of the examined system. The graphic economy is achieved and thus the diagrams are effectively presented to the users. With the use of different views, Secure Tropos does not concentrate

vast amounts of information in the same model, but distinguishes information according to the focus of each part of the analysis. Thus, the principle of Dual Coding is fully satisfied.

Principle of Cognitive Fit. Secure Tropos modelling language is not provided in two versions, as it is suggested by this principle. There is the requirement that the language should be provided in different versions, covering mainly the level of expertise of users, as due to its wide applicability, it is used by students, IT security experts, project managers, and also simple users. Thus, the principle of Cognitive Fit is not satisfied.

From the above analysis, it is revealed that Secure Tropos modelling language fully satisfies four out of nine principles, four of them are partially satisfied, while one principle is not satisfied at all. These results can be used in order to better improve the language, focusing in the revision of specific elements which can contribute to the overall communication of the language with its users.

6 Conclusions

The effectiveness of a methodology to efficiently communicate its content with the users is of equal importance to the semantics of it. In this paper we evaluate a security and privacy requirements engineering methodology, namely Secure Tropos, based on the most well-known theory, the Physics of Notation, which has been synthesised from theory and empirical comparison and can be used for the evaluation, comparison and improvement of visual notations. Our qualitative analysis resulted in valuable lessons learned, which are thoroughly discussed in Sect. 4, and can also be applied to other security and privacy requirements engineering methodologies. This application, which is one of our future works, will allow us (i) to evaluate them and proceed to comparisons among them, and (ii) to develop guidelines for the improvement of their visual syntax.

Moreover, empirical analysis is also another future step, in order to identify to what extend the proposed outcomes of the analysis of this paper can improve the communication between the analysts and end users. The users have to be distinguished between experts and novices and the aim is to record their perception regarding the **design goals**, such as simplicity, aesthetics, expressiveness and naturalness, and also, regarding **cognitive effectiveness**, such as speed, ease, and accuracy.

Finally, in order to further strengthen the validity of our results, external practitioners will be involved in the study. This could substantially raise the subjectiveness of the evaluation part of this research.

References

1. Alexander, C.: Notes on the Synthesis of Form, vol. 5. Harvard University Press, Cambridge (1964)
2. Avison, D., Fitzgerald, G.: Information Systems Development: Methodologies, Techniques and Tools. McGraw Hill, Maidenhead (2003)
3. Bar, M., Neta, M.: Humans prefer curved visual objects. *Psychol. Sci.* **17**(8), 645–648 (2006)
4. Bertin, J.: *Semiology of Graphics: Diagrams, Networks, Maps* (1983)
5. Blackwell, A., Green, T.: Cognitive dimensions of notations resource site (2009). <http://www.cl.cam.ac.uk/afb21/CognitiveDimensions>
6. Butler, J., Holden, K., Lidwell, W.: *Universal Principles of Design: A Cross-Disciplinary Reference* (2003)
7. Caire, P., Genon, N., Heymans, P., Moody, D.L.: Visual notation design 2.0: towards user comprehensible requirements engineering notations. In: 2013 21st IEEE International Requirements Engineering Conference (RE), pp. 115–124. IEEE (2013)
8. DeMarco, T.: *Structured Analysis and System Specification*. Yourdon Press, Upper Saddle River (1979)
9. Gehlert, A., Esswein, W.: Toward a formal research framework for ontological analyses. *Adv. Eng. Inform.* **21**(2), 119–131 (2007)
10. Goolkasian, P.: Pictures, words, and sounds: from which format are we best able to reason? *J. Gen. Psychol.* **127**(4), 439–459 (2000)
11. Grady, B.: *Object-Oriented Analysis and Design with Applications* (1994)
12. Green, T.R.G., Petre, M.: Usability analysis of visual programming environments: a ‘cognitive dimensions’ framework. *J. Vis.Lang. Comput.* **7**(2), 131–174 (1996)
13. Green, T.R.: Cognitive dimensions of notations. In: *People and Computers V*, pp. 443–460 (1989)
14. Gurr, C.A.: Effective diagrammatic communication: syntactic, semantic and pragmatic issues. *J. Vis. Lang. Comput.* **10**(4), 317–342 (1999)
15. Harel, D.: On visual formalisms. *Commun. ACM* **31**(5), 514–530 (1988)
16. Harel, D., Rumpe, B.: Meaningful modeling: what’s the semantics of “semantics”? *Computer* **37**(10), 64–72 (2004)
17. Hitchman, S.: The details of conceptual modelling notations are important—a comparison of relationship normative language. *Commun. Assoc. Inf. Syst.* **9**(1), 10 (2002)
18. Irani, P., Ware, C.: Diagramming information structures using 3d perceptual primitives. *ACM Transactions on Computer-Human Interaction (TOCHI)* **10**(1), 1–19 (2003)
19. Kalyuga, S., Ayres, P., Chandler, P., Sweller, J.: The expertise reversal effect. *Educ. Psychol.* **38**(1), 23–31 (2003)
20. Kim, J., Kim, M., Park, S.: Goal and scenario based domain requirements analysis environment. *J. Syst. Softw.* **79**(7), 926–938 (2006)
21. von Klopp Lemon, A., von Klopp Lemon, O.: Constraint matching for diagram design: qualitative visual languages. In: Anderson, M., Cheng, P., Haarslev, V. (eds.) *Diagrams 2000*. LNCS (LNAI), vol. 1889, pp. 74–88. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44590-0_11
22. Lankhorst, M.: *Enterprise Architecture at Work: Modelling, Communication and Analysis*. The Enterprise Engineering Series. Springer, Heidelberg (2009)

23. Larkin, J.H., Simon, H.A.: Why a diagram is (sometimes) worth ten thousand words. *Cogn. Sci.* **11**(1), 65–100 (1987)
24. Mellado, D., Blanco, C., Sánchez, L.E., Fernández-Medina, E.: A systematic review of security requirements engineering. *Comput. Stand. Interfaces* **32**(4), 153–165 (2010)
25. Moody, D.: The “physics” of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Softw. Eng.* **35**(6), 756–779 (2009)
26. Moody, D.L.: Complexity effects on end user understanding of data models: An experimental comparison of large data model representation methods. In: *ECIS 2002 Proceedings*, p. 10 (2002)
27. Mouratidis, H.: A natural extension of tropos methodology for modelling security (2002)
28. Mouratidis, H., Argyropoulos, N., Shei, S.: Security Requirements Engineering for Cloud Computing: The Secure Tropos Approach. *Domain-Specific Conceptual Modeling: Concepts, Methods and Tools*, pp. 357–380. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39417-6_16
29. Nordbotten, J.C., Crosby, M.E.: The effect of graphic style on data model interpretation. *Inf. Syst. J.* **9**(2), 139–155 (1999)
30. Opdahl, A.L., Henderson-Sellers, B.: Ontological evaluation of the uml using the Bunge-Wand-Weber model. *Softw. Syst. Model.* **1**(1), 43–67 (2002)
31. Pavlidis, M., Islam, S.: Sectro: a case tool for modelling security in requirements engineering using secure tropos. In: *CAiSE Forum*, pp. 89–96 (2011)
32. Purchase, H.C., Carrington, D., Allder, J.A.: Empirical evaluation of aesthetics-based graph layout. *Empirical Softw. Eng.* **7**(3), 233–255 (2002)
33. Shanks, G., Darke, P.: Understanding corporate data models. *Inf. Manage.* **35**(1), 19–30 (1999)
34. Shanks, G., Tansley, E., Weber, R.: Using ontology to validate conceptual models. *Commun. ACM* **46**(10), 85–89 (2003)
35. Siau, K., Cao, Q.: Unified modeling language: a complexity analysis. *J. Database Manage. (JDM)* **12**(1), 26–34 (2001)
36. Wiegmann, D.A., Dansereau, D.F., McCagg, E.C., Rewey, K.L., Pitre, U.: Effects of knowledge map characteristics on information processing. *Contemp. Educ. Psychol.* **17**(2), 136–155 (1992)
37. Winn, W.: Encoding and retrieval of information in maps and diagrams. *IEEE Trans. Prof. Commun.* **33**(3), 103–107 (1990)
38. Yu, E., Liu, L., Mylopoulos, J.: A social ontology for integrating security and software engineering. In: *Integrating Security and Software Engineering: Advances and Future Actions*, pp. 70–105 (2006)

Security Requirements Elicitation and Modelling

What Users Want: Adapting Qualitative Research Methods to Security Policy Elicitation

Vivien M. Rooney and Simon N. Foley^(✉)

IMT Atlantique, LabSTICC, Université Bretagne Loire, Rennes, France
{vivien.rooney,simon.foley}@imt-atlantique.fr

Abstract. Recognising that the codes uncovered during a Grounded Theory analysis of semi-structured interview data can be interpreted as policy attributes, this paper describes how a Qualitative Research-based methodology can be extended to elicit Attribute Based Access Control style policies. In this methodology, user-participants are interviewed, and machine-learning is used to build a Bayesian Network based policy from the subsequent (Grounded Theory) analysis of the interview data.

1 Introduction

A challenge in eliciting security requirements is properly understanding the needs of the user. The requirements for security mechanisms are all too often determined by technical experts who interpret user needs through the lens of their own technical experience, and which may be at variance with the requirements of the end-user. While user-centered security [2,33] can help to provide usable security mechanisms, the tendency is to focus more on ensuring that the mechanism provides a good user-experience, and not so much on discovering what the user, as a person, actually needs from security [12]. When a security mechanism does not match the user needs, the user either figures out some circuitous way to bypass the security control or else avoids using the system properly [3,8].

The goal of security experts is ensuring security, and one of the many challenges is being able to encompass users' important personal values [12]. Furthermore, the elicitation of user needs remains a stumbling block in the security community. Capturing the subtleties of user requirements is challenging [18]. Eliciting needs and requirements from people is not a new problem. Sociologists and applied psychologists have spent decades attempting to understand people's experience and, more recently, their experience of technology.

Qualitative research methods have evolved as a means to systematically elicit the needs of users. Understanding an experience such as chronic illness requires a method that can delve into the subtle minutiae of living with such a condition, and explore the meaning of that experience, for example, the significance of disclosure in the workplace [10]. The methods can be applied to technology, to find out, for example, about the perspective that mothers' have on mobile communication technology [26]. This knowledge provides us with new ways of understanding, for instance, what the disclosure of a chronic illness means to

people, whether that is about control of their body, or the consequences for their social identity. We can encompass the subtle meanings of events or artefacts that might otherwise remain unremarked.

We are interested in understanding how Qualitative Research techniques might be used in a systematic way to elicit security requirements from users. However, the challenge of understanding the human experience of technology is such that there can be no ‘silver bullet’ for eliciting requirements for all possible scenarios. We therefore limit our study to the elicitation of attribute based policies. Our motivation for focusing on these policies in this paper comes from previous research [16] which suggested that Grounded Theory, a qualitative data analysis technique, might be used to help uncover attributes of a Qualitative Bayesian Network that in turn describes the elicited policy. In this paper we build on this by considering the challenges of integrating it as part of a broader qualitative methodology for eliciting requirements.

This paper is organized as follows. Section 2 considers existing literature on eliciting security needs and discusses the challenges to carrying out qualitative research in practice. Section 3, using a simple use-case, describes how a Qualitative Researcher would conventionally approach understanding a user’s needs, in this instance, by using semi-structured interviewing and Grounded Theory analysis. Grounded theory as a research approach is a systematic and in-depth process. While qualitative research is demanding and time-consuming, it is widely accepted as a valid means of gaining an insightful understanding of the user’s experience and needs around technology [1]. Section 4 describes how machine learning of the interview transcripts marked up during Grounded Theory analysis can be used to generate a Bayesian network that provides a model security policy of the requirements. Encoding the policy as a probabilistic network recognizes that the policy needs of the user gathered during elicitation may be approximate and can be further learned based on subsequent user behavior. Having illustrated how a qualitative researcher would approach an elicitation exercise, Sect. 5 considers the lessons learned and considers how a technical person, who is not a Qualitative Researcher, might approach the methodology.

2 Related Research

Research on security requirements engineering has tended to focus on the engineering lifecycle of capturing/analyzing security requirements, through to system development, and with respect to functional requirements. Approaches such as SecureUML [4] can help to model and analyze aspects of understood requirements, while threat driven approaches [14] use threat and countermeasure use-cases to help understand and drive requirements. Methodologies such as Secure Tropos [22,23] provide a goal-oriented modeling approach to capture and analyze socio-technical security requirements, providing engineering support from early requirements through to detailed design. With respect to security requirements, their emphasis is more about requirements capture through the use of expressive modeling languages, and less about *how* to draw out the needs of the

individual user. In this paper we focus on the elicitation of security needs from the user as a human being. We consider how the requirements engineer can come to learn about these needs by interacting with the user. Intuitively, elicitation occurs before, and provides input to, requirements capture, for instance, before the early requirements phase in Secure Tropos [22, 23]. We focus on requirements for security policies constructed in terms of attributes, as might be used in an attribute based policy. Whether this elicitation technique can be applied to more general requirements capture is a topic for future research.

Our position is that Grounded Theory analysis [9] of semi-structured interviews [20] can provide a methodological basis for eliciting these policy attributes. Grounded Theory is a qualitative research method commonly used in social sciences for generating theories demonstrably grounded in data, hence avoiding the imposition of a priori assumptions about the problem domain. Data can range from transcripts of interviews to recordings of interaction. Existing computer-assisted qualitative data analysis software/Grounded Theory analysis tools, such as NVIVO, provide editing and syntactic analysis, but do not provide semantic modeling. Qualitative methods are often used in usability security studies to better understand the user experience of security. For example, [32] examines the experience of regret in social media use.

In Computer Science, qualitative methods have been used to help elicit requirements in Software Engineering [28] and compliance [6]. In the security domain, for example [2, 13, 15, 25], Grounded Theory has been used to help understand user behaviour as part of security system design. Personal privacy/security assistants such as [7, 19] help users decide policy, however by taking a structured/questionnaire based approach to eliciting policy they pre-judge the attributes that are important, in contrast to the approach taken in this paper.

Qualitative Methods are coming to the fore as a systematic approach for uncovering emergent security requirements. In [27] Grounded Theory is used in conjunction with fault-trees as a methodological means to identify emergent threats and thereby discover unknown knowns. In [18], interviews and focus groups were carried out in order to understand Grid access control needs, and an access control language was developed to support these requirements. Studies have used an ethnomethodological approach to elicit privacy requirements for mobile applications [5, 30]. Although these studies help uncover security and privacy needs, the difficulty lies in taking these needs, elicited in qualitative data, and rendering them into requirements that can be directly implemented by developers. The next step is transforming these needs into actionable requirements. The methodology proposed in the following sections addresses this challenge.

3 A Qualitative Research Perspective on Elicitation

3.1 A Use-Case

One of the authors, an Applied Psychologist, conducted a qualitative study to research how people make sense of photograph sharing using their camera phone.

In this context, privacy and identity are closely related, and the control of personal identifying information is fundamental [29]. The use of mobile phones has created social norms in relation to acceptable behaviour and habits, and normative behaviour continues to evolve [17]. Understanding social norms around photograph sharing means developing an understanding of how people make sense of this activity in their evolving social world. Conventional photo-sharing security controls can be too coarse grained for user needs and users work around them in order to achieve their shared goals [3]. The aim of this use-case is to go beyond these existing security requirements, exploring individual values, beliefs and experiences. This will facilitate developing an understanding of their inter-play and thus why, and how, the nuances of personal preferences are formed.

While the outcome of such a study builds an understanding of people's relationship with technology, from the perspective of user-centered design, it is also relevant to eliciting their needs/requirements for security. In this section we sketch this elicitation process as a form of qualitative research. Section 4 will describe how security policies can in turn be generated from this process.

The focus of the current paper is methodological, hence the original qualitative study will be described briefly. This is detailed elsewhere [16]. The background, in brief, is that seven interviews were conducted, with a duration ranging from 17 to 54 min, generating a total of 226 min of data. Interview number 7, on which the use case is based, generated a total of 51 codes during the analytic process. The use case is based on a subset of those codes.

3.2 Qualitative Research

The Qualitative Research approach to finding out what people need and want in a particular set of circumstances is to understand how they make sense of that situation. How people make sense of a situation, their unique perspective, is based on the whole of their experience. Experience itself is a complex concept. A person's past, present and anticipated future, are part of their experience. Experience is comprised of components that are physical, social, intellectual and emotional. Given that one person's experience can include these intersecting and non-linear aspects, then the breadth and depth of that experience presents a challenge for understanding how they make sense of a particular situation.

Understanding experience demands methodological resources in terms of skills and time. Resources are finite, and researchers adapt accordingly. Thus, at the outset of a research project, a simple yet important question is asked: what is it that you want to find out. For example, from a social psychological perspective, a question might be: how is a person sharing photographs. If I decide to approach the research by compiling a questionnaire, then I will have answers that reflect a particular set of questions. This set of questions will be the same for all participants, and as such, curtails the scope of possible answers. However, with this method, the advantage is in the large number of possible participants. If I decide to approach the research in a different way, I have scope to answer different questions. For example, I can find out, not alone how people share photographs, I can also find out how they make sense of their sharing practices,

about the values that underpin their reasoning, and how those values might have developed. With the latter approach to research, a qualitative approach, it is possible to understand the breath and depth of a person's experience, relevant to a particular set of circumstances.

Qualitative Research is characterised by a diverse and evolving methodology, including ethnography, participant observation, and semi structured interviewing. Analytic methods include Grounded Theory and Discourse Analysis. Methodological decisions are underpinned by particular epistemologies [31]. One position is Social Constructionism, taking the view that humans actively create their world in social interaction. Researchers working from this perspective would argue that research is not objective, rather what emerges is interpretative. Approached in this way, semi-structured interviewing is considered to be a creative engagement where the unexpected can emerge in the dialogue between interviewer and participant. In contrast, a structured interview utilises the same schedule for all participants, and is similar to a questionnaire (Smith, 1995) curtailing what is discussed. Thus, with semi-structured interviewing, rather than structuring the dialogue to fit a preconceived format, it is possible to explore what is unique for each participant. The data that emerges is analysed to understand the process of making meaning.

3.3 Eliciting via Semi Structured Interviewing

The skill of conducting semi structured interviews is in itself the subject of scholarly study [20]. In the current research, the example being used for methodological purposes is interviewing in order to find out about photograph sharing practices. This means, as discussed above, delving into the personal values and beliefs that inform choices. With the aim of developing an in-depth understanding of the unique perspectives that individuals bring to making sense of photograph sharing, semi structured interviews are an ideal way to proceed.

Temporal structure of an interview. The interview schedule is a guide for the researcher. The interview follows a temporal structure: the present, the future and the past. This facilitates the participant talking about and reflecting on the subject matter. Discussing the present can be an easy way to begin a dialogue to ensure that, as far as possible, the participant is comfortable with the process and subject matter. During interview, the participants decide on, and choose, the particular matters discussed and the extent that they wish to go into detail. This may include decisions made in the past, whether they now have a different perspective on what they did, and why that may have changed over time. Hypothetical questions related to the area of interest facilitate the participant in considering their approach to the subject matter, and may prompt recall of particular incidents, and conversation about what is, and was, relevant to their decision making. Meaningful past events may have prompted participants to reflect on the subject matter in depth. Such events and reflections can provide valuable insights into the participant experience, and their psychological world.

Develop the interview schedule. This directs the dialogue with the participant. A sample is provided in Fig. 1. The schedule is not referred to during the interview, to avoid disrupting the flow of the conversation. This process also reflects the semi-structured nature of the interview, as unexpected avenues of discussion raised by participants can become an alternative focus, and this is facilitated by the use of prompts. A sample is provided in Fig. 2.

- VR: Have you used your camera phone to take photos of family and friends?
 VR: Have you kept any of those photos? Can I see some?
 VR: Would you give copies of your photographs to someone else? Say that you had a photo of a friend in your camera and another person saw it, but didn't know them, and wanted a copy. Would you give them a copy?
 VR: Would you give me a copy? Would you give anyone a copy? How am I different?
 VR: Do you think it would be the same if the photograph was of people that neither of you knew, say you took a photo of some famous landmark, like the leaning tower of Pisa, and the photo turned out to have people in it, so you could see them clearly. Would you give those photos to anyone who wanted them?
 VR: Would you take a photo of someone you knew if they didn't know what you were doing? Say you saw a friend on the street but they didn't see you, would you think it is ok to take their photo?
 VR: What about if someone, maybe a friend or a family member, was at your house and say, fell asleep, would you take their photo?
 VR: Would you feel you should show it to them or not? Give them a copy if they wanted it?
 VR: Has anyone ever taken a photo of you when you didn't know about it? What do you think about that? Would you mind what they did with the photo? Say if they used it in an advertising campaign? Why?
 VR: Have you taken photos that you felt were intrusive, maybe the people didn't know what you were doing, maybe they were asleep, maybe they couldn't stop you?
 VR: Would you take a photo of someone you didn't know on the street, say you liked their haircut or shoes and wanted to copy the hair or something, would you do that?
 VR: Why not? Would it be different if they couldn't see you? Would you mind if they saw you?
 VR: I am wondering about taking photos of people who don't have cameras or access to cameras. Say you were out in a place, maybe a city, and saw people living on the street, would you take their photos?
 VR: Is there anything else you would like to add?

Fig. 1. Interview schedule excerpt

Address informed consent. Provide information on the research project and methodology, including its qualitative nature using semi structured interviewing and Grounded Theory. Explain anonymity and deidentification, that anything said during interview can be excluded if the participant wishes, and the freedom to end participation at any time, without consequence. Provide contact details of the researcher, offer a copy of the transcript, explain that verbatim quotations

might be used in academic publications. Explain transcription and analysis, including that only the researcher may access the audio recording, which would be deleted following analysis, and that the interview would cease at any time if they chose to do so. Informed consent is regarded as a process in Qualitative Research, and the preceding summary is for illustrative purposes.

VR: any idea why?
 VR: that's interesting
 VR: hadn't thought of it like that
 VR: I wonder why?
 VR: could you tell me more about that?
 VR: could we talk about that?
 VR: I'd like to know more about that

Fig. 2. Interview prompts

3.4 Data Analysis

Transcribe the audio recordings of interviews. The goal of transcription is the production of an account of an interview that is manageable, readable, and amenable to the analytic method [20,24].

Code Data. Grounded Theory techniques for data analysis include the coding data at various levels, the use of constant comparison during coding, the generation of categories, and Memo writing. Initial coding can be a line by line process, whereby labels are applied to text, to assign and encapsulate meaning. A coded piece of text could be a phrase, a sentence, or a few sentences [9].

The process of coding is described in Memos, as are ideas on the direction of the analysis. Generating categories means grouping codes together to reflect the emerging analysis.

Initial categories. The following categories were generated by the analysis.

- Control of the images
- Entitlement to photographs for the person in them
- Images of self
- Privacy
- Sharing of images
- Photographs of strangers contrasted with friends
- Trust
- Treating others as I wish to be treated (empathy)

Analysis is a flexible process, and categories may be merged or refined, reflecting ideas for theoretical development. Links between categories, such as similarity and difference, are explored and described. All of the foregoing steps are recorded in Memos, which are rewritten and expanded iteratively during the research process. Qualitative research using Grounded Theory methods is time consuming, requiring an intensive engagement with the data. The preceding summarises this for illustrative purposes.

4 Moving from Grounded Theory to Attribute Policies

Grounded Theory analysis, through coding, identifies a range of code phenomena that can be considered to provide an interpretation, or semantics, of the syntax of the interview text. It is argued [16] that these codes might be treated as discrete probabilistic variables, representing the probability of the occurrence of the phenomena of interest; however, [16] does not consider how it can be generated in practice. In this paper we propose a \LaTeX based notation used by the Analyst to mark up interview text¹ with details about the codes and their dependencies. The marked-up script provides a meaning for its content, and machine learning is used on this marked-up data to build a Bayesian Network based policy that represents the relationships between the identified phenomena. Coding identifies a variety of attributes, as probabilistic variables, including those representing the characteristics of entities of interest, and actions and decisions. For example, code (attribute) `child` means that a photograph contains a child, while code `share` corresponds to the action of sharing a photograph.



Fig. 3. Sample Bayesian Network policy

Figure 3 gives a fragment of a policy for the photograph-sharing use-case. Variables `suffering` and `child` are observed; for example, indicating the presence of corresponding image tags in a photograph, where `child` means the scene contains a child, and so forth. Latent attributes `vulnerable` and `share` are inferred and `share` represents the decision to share the photograph in question, potentially with a family member, or otherwise. A policy query is interpreted as: *given a collection of observations that describe attributes related to some proposed action, then what is the probability of a given decision?* For example, what is the probability that it is OK to share a photograph depicting a child (tag) in a public place (tag) with a family member? In eliciting a policy, one identifies these variables along with associated probability distributions as follows.

4.1 Line by Line Coding

The observation of a phenomena, denoted by the code `v:c`, uncovered during a Grounded Theory analysis, is represented by a state `c` of a discrete random variable `v`. This observation is identified by marking-up, using a line by line code, in the dialogue transcript in which the phenomena is observed. We assume that

¹ \LaTeX was chosen for expediency.

each code uncovered during a Grounded Theory analysis corresponds to a state (identified as *c*) of some random variable (identified as *v*) and a coding markup (in L^AT_EX) `\qaCode{v:c}{text}` specifies the observation of a phenomena, as state *c* of a random variable *v*, in the given transcript text. For example, observations are noted about the public and private sharing of photographs:

```
\qaCode{share:private}{protecting people's dignity and privacy and things like that} [...] Its probably more straightforward when its family and friends but if you're using it in a \qaCode{share:public}{public context to serve a different purpose, I think then maybe its a bit harder to weigh it up,} [...]
```

and this is typeset in L^AT_EX as:

Answer: protecting people's dignity and privacy and things like that [...] its probably more straightforward when its family and friends but if you're using it in a public context to serve a different purpose, I think then maybe its a bit harder to weigh it up,	<div style="color: blue; margin-bottom: 10px;">share:private</div> <div style="color: magenta;">share:public</div>
--	--

A predefined taxonomy of policy variables/states is not required: it is the coding during Grounded Theory that surfaces the variables/states relevant to the policy.

During coding, a phenomena *v:c* is routinely characterized as a tautology, and therefore, for simplicity, we assume that its complementary state is identified syntactically as *!v*. Furthermore, many phenomena are binary in nature, and when no ambiguity arises, we assume that the state and its corresponding variable use the same identifier and drop reference to the random variable in marked-up text. For example, code *child* denotes a state of variable *child*, reflecting the observation of phenomena of the presence of a child in a photograph:

```
\qaCode{!child}{I dunno, pictures of kids, they're bordering on something else, they're, I don't know, I don't know if I'd take pictures of [...]}
```

and code *!child* denotes an observation that a child is not present in a photograph. Thus, probabilistic variable *child* has literal states *child* and *!child*, while probabilistic variable *share* has states *sharepublic*, *shareprivate* and *!share*.

4.2 Observing Collections of Phenomena

During the coding process, the analyst may group any number of codes together using code conjunction (+) and disjunction (,) operators in order to specify a simultaneous observation of phenomena in the transcript. For example,


```
\qaCode{child+vulnerable }{[...] kids don't tend to have that ability
to be able to tell if it was a right or wrong thing or what they
think or what they feel , no I don't think I'd take pictures of
anyone's children , of any children . }
```

is the observation that the participant is relating a child as a vulnerable person in the photograph. In coding this simultaneous observation of the phenomena, no assumption is made about the nature of any statistical dependency between the variables, other than the observation of a simultaneous occurrence of states child and vulnerable.

The analyst may also assert there is no known relationship between phenomena, but for convenience, wishes to group the codes together using disjunction. For example,

```
\qaCode{share:public,share: private }{it could be public or private }
```

Inclusion of one code within the scope of the text of another is considered to define a simultaneous observation of phenomena. For example, during a discussion about sharing photographs privately, the participant remarks

```
[...] \qaCode{share:private }{[...] because there's a line that you
don't cross when it comes to \qaCode{vulnerable}{protecting people's
dignity and privacy} and things like that and I think the difficulty
sometimes is trying to weigh that up.}
```

reflects that, in addition to the observation of the phenomena of sharing photographs, there is a simultaneous observation of privately sharing photographs of vulnerable individuals.

Axial-coding, which is used in Grounded Theory to relate codes (categories and concepts) to each other via a combination of inductive and deductive thinking, organizes codes according to categories, for example,

```
\qaCode{place>developingcountry }{for instance pictures taken in
developing countries of starving children and [...]}
```

the axial code `place>developingcountry` specifies an observation about the code phenomena `developingcountry` related to the code category `place`. While categories may be associated with multiple codes, a category may also define a code in itself. For example,

```
\qaCode{vulnerable>suffering }{[...] I think to me it would be
exploiting that person really and considering their circumstances, its
almost like you're taking, sort of dehumanizing that person, almost
objectifying them sort of, so in a sense you're homeless, you're on
the street , }
```

In this case, markup with the code `vulnerable>suffering` specifies the simultaneous observation of codes `vulnerable` and `suffering` in the text, and this is interpreted as the code expression `vulnerable`, (`vulnerable+suffering`).

The analyst uses the codes to reflect the observations concerning phenomena. When there is ambiguity then we assume that the analyst will either use the codes to reflect this ambiguity or will seek to provide additional context in order to eliminate ambiguity. For example, a participant is unsure whether or not he would be comfortable in privately sharing a photograph of a child,

```
\qaCode{share:private,share:public,!share}{ \qaCode{child,!child }{
sometimes I'll share a photograph of a child and sometimes not } }
```

However, the goal of the semi-structured interview and subsequent Grounded Theory analysis is, in this case, to uncover the phenomena that characterize sharing of photographs, and, therefore, it is anticipated that such ambiguity should be avoided.

Similarly, rather than using weighting schemes, it is assumed that sufficient context is provided during coding in order to clarify the significance of a phenomenon. For example,

```
\qaCode{share:public+adult}{I'd nearly always share a photograph
containing an adult }
```

During the interview, a context is elicited

```
\qaCode{!share+adult+suffering }{[...] but not if the person was
suffering }, \qaCode{share:private+adult+suffering}{but perhaps it
would be OK if done privately }
```

4.3 Describing Dependencies Between Phenomena

During the process of analysis, dependencies between the codes/random variables are inferred by the analyst. For example, the analyst identifies a dependency between codes `vulnerable` and `child` and decides that the value of `child` influences `vulnerable` in some way. Similarly, the analyst decides that the decision on sharing is influenced by `vulnerable` and whether it is with a family member.

The text of the dialogue is marked up to reflect these phenomena, though it need not be tied to any particular part of the dialogue, as it is a theory that emerges over the entire transcript. These code dependencies, in turn, are used to define the variable dependencies in the generated Bayesian Network policy. For example, the markup `\qaDep{vulnerable}{share}` specifies dependency `vulnerable→share`. During the course of the Grounded Theory analysis of the photograph sharing interviews, dependencies were identified and marked up in the \LaTeX source, and, for the example in this paper, these included the following.

```
\qaDep{child}{vulnerable}
\qaDep{adult}{vulnerable}
\qaDep{suffering}{vulnerable}
\qaDep{vulnerable}{share}
\qaDep{family}{share}
```

4.4 Learning Policies

The marked-up transcript identifies (multiple) occurrences of phenomena (variables) and their dependencies and these are used to construct the structure for the Bayesian Network policy given in Fig. 3. Consider the typeset fragments of marked up interview text in Fig. 4; the observed codes and code-dependencies marked-up in the \LaTeX source during analysis are typeset in the margin.

[...]

VR: *do you think if you were out on the street and there was a homeless person, maybe asleep, and if you looked at that person and thought, I want to take a photograph of that because I want to make a point about it, I want to use it sometime, do you think you'd take that kind of photograph*

Bob: *no, I wouldn't, I think it's just probably because, I think to me it would be exploiting that person really and considering their circumstances, it's almost like you're taking, sort of dehumanising that person, almost objectifying them sort of, so in a sense you're homeless, you're on the street, so therefore, I can take a photograph and the intention might be to publicise the issue, but in actual fact in doing that you're also sort of putting a negative spin on it as well, but I feel the same way about, for instance, pictures taken in developing countries of starving children and stuff like that, and tee shirts with logos from different organisations because you might argue on one hand that you have to advertise for the cause and it's important to raise peoples' consciousness about issues like that, but in saying that there's also something that I find a bit disturbing in that because there's a line that you don't cross when it comes to protecting people's dignity and privacy and things like that and I think the difficulty sometimes is trying to weigh that up.*

It's probably more straight forward when it's family and friends but if you're using it in a different context, to serve a different purpose, I think then maybe it's a bit harder to weigh it up,

[Excerpt of interview 2 with Bob]

vulnerable

suffering→vulnerable
(vulnerable)+not-share

child→vulnerable
((child,suffering)+vulnerable)

not-share+(suffering,child)
vulnerable+not-share

family→share
family+share,not-family+not-share
vulnerable→share

Fig. 4. Typeset interview

Each code markup in the transcript is a code-expression corresponding to the observation of a collection of phenomena, and we store each markup as a disjunctive normal form dataset of observed codes. Figure 3 provides an example of the phenomena observed in the above, along with the generated dataset. Note that absence (“-”) of a phenomenon observation means that nothing is known about that code and it does not necessarily mean the complement of the state. Recording (child+vulnerable) indicates the phenomena of a vulnerable child has been observed, but no observation is made about sharing at that point in the transcript. Equally, recording (share+family) indicates the willingness to share a photo with a family member, but with no statement made about other variables at that point.

	child	family	share	suffering	vulnerable
vulnerable	-	-	-	-	vulnerable
(vulnerable)+!share	-	-	!share	-	vulnerable
((child,suffering)+vulnerable)	child	-	-	-	vulnerable
!share+(suffering,child)	-	-	-	suffering	vulnerable
vulnerable+!share	-	-	!share	suffering	-
family+share, !family+!share	child	-	!share	-	-
	-	-	!share	-	vulnerable
	-	family	share	-	-
	-	!family	!share	-	-

Fig. 5. Selected observations and generated DNF dataset

Intuitively, each line of the dataset in Fig 5 represents an access control rule about sharing, based on attributes (codes) discovered during analysis. However, we cannot treat this dataset as exhaustive: no matter how expressive a phenomena-coding markup language might be, every possible sharing combination cannot be explicitly discussed during an interview. It is therefore necessary to estimate the gaps in the policy rules by inferring the probabilities for the variables, including the transitional probabilities. For ease of exposition in this paper, we took a somewhat promiscuous view of access control, whereby the Expectation Maximum (EM) learning algorithm [21] is used to maximize the variable probabilities. This means that a user is assumed likely to share photographs so long as it not inconsistent with their policy (a probabilistic default share). Learning, based on a probabilistic default not-share is a topic for future work. The appendix gives the details of the generated Bayesian Network.

In the interview fragment above, the participant predominantly speaks of photographs depicting suffering, children and vulnerability, and therefore, based on the markup/observations in Fig.5, the probabilities of these events in the policy are high. In the full interviews, the participants spoke of other occasions when the photograph subject was not considered vulnerable, leading to a lower likelihood of vulnerability in the policy. Based on the EM-learning of the

observations in Fig. 5, the conditional probability table calculated for the latent variable `share` in the above example is:

	family		!family	
	vulnerable	!vulnerable	vulnerable	!vulnerable
share	0.8148	1.0	0.0964	0.0
!share	0.1852	0.0	0.9036	1.0

In our example, the learning of probabilities may be open to question given the small number of observations in the dataset of interview markup provided in the fragment above. However in practice, qualitative analysis identifies not just the existence of a code, but marks up every occurrence of that code throughout the interview and thereby ensuring a larger and more effective learning dataset. Intuitively, the more the participant touches on a phenomenon in the interview, the more the code is observed/marked-up and thus, the more likely it is considered to occur. Whether it is methodologically qualitatively sound to assume that the more a participant touches on a topic during an interview, then the more relevant that topic is, is open to discussion. Designing a more expressive markup language whereby the analyst can override/weight significant phenomena is a topic for further work.

A \LaTeX package was implemented that generates a Bayesian network as described above. Based on the \LaTeX markup, the package generates the observed phenomena dataset and the Samlam [11] EM learning implementation uses this to generate the probabilities for the Bayesian Network defined by the dependencies in the markup.

4.5 Policies in Practice

In this paper a policy is represented as a Bayesian Network, reflecting the approximate nature of the data gathered and analyzed during elicitation. In one prototype application for these policies, a ‘clippy’ style Android photograph sharing assistant was implemented that uses the elicited policy to provide security advice to the user as to whether it is safe to upload and share a given photograph on Google Picasa; it was built using the Netica-J API for Android.

The Android camera phone enables the user to tag camera JPEG images with the attributes identified during the elicitation process. These tags provide observations for variables in a policy query, while the values for other policy variables can be based on the outcome of the EM-learning carried out during policy elicitation. The outcome of a query is the probability of the `share` variable: during a photograph upload, an alert is generated if the share probability is below a user-defined threshold, and the user is given the option to override. If the user decides to override the advice, or to add previously unseen tags, then this new sharing behavior can be further learned and the policy dynamically updated.

4.6 Methodological Threats

With the aim of developing an in-depth understanding of the subtleties and nuances of participant experience, the research methods of semi-structured interviewing [20], in conjunction with Grounded Theory [9] analysis were selected as the most suitable to answer the research question. The scope of the qualitative study was subject to practicalities, such as time frame and available resources. The time frame was a period of six months, with a resource of one researcher available on a part-time basis. With the foregoing constraints, the number of possible participants was deemed to be between 5 and 10. In the event, 7 participants were interviewed. The recruitment, data collection and analysis were conducted by a single researcher. Epistemologically, Social Constructionism underpins the original qualitative research, and is coherent with the interpretivist approach taken in the data analysis [31]. Hence, the results make no claim to objectivity, or to be applicable to the general population. Rather the emergent analysis is an in-depth and subjective exploration of the experience of a small group, on which the use-case is derived.

5 Discussion and Conclusion

In this paper we consider how techniques that are used in Applied Psychology to understand a person's feelings and needs might provide a means to elicit their security needs. While it is a truism that Qualitative research techniques could be used to achieve this, the contribution of this paper is to map the activity into something actionable, that is, provide a means to generate a machine-interpretable security policy. Recognising that the codes uncovered during a Grounded Theory analysis of semi-structured interview data can be interpreted as attributes for an attribute-based access control policy, the paper describes how the Qualitative Method proceeds, from interviewing the user-participant, to analysing and uncovering the codes, and to mapping these codes to probabilistic variables in a Bayesian Network that provides the final policy.

One of the key contributions of the paper is the demonstration of how Grounded Theory can be used to identify policy attributes and their relationships. We chose to use a Bayesian Network as the policy model in order to support incompleteness in elicitation. Mapping these attributes to a particular ABAC model, for example XACML, is a topic for future research. We believe that the Qualitative approach in Sect. 3 could assist in eliciting attributes for more general requirements, such as SI* requirements [22]; however, the extent to which requirements could be learnt using a strategy similar to Sect. 4 is an open question.

Grounded Theory analysis of semi-structured interview data is effective at uncovering individuals' needs and wants, however, by virtue of what it attempts to uncover, it is a costly and time-consuming activity, and requires a high degree of skill to carry out properly. As a technical person, a security requirements engineer is unlikely to have the requisite skills, nor the luxury of resources, to be able to conduct such an in-depth study in order to obtain the user policies.

However, we argue that a more conventional elicitation of user requirements will not necessarily uncover the user's true needs to the same degree.

We are exploring how the Qualitative method described in Sect. 3 could inform a more lightweight methodology for policy elicitation that could be used by a Requirements Engineer in a cost-effective manner. For example, rather than conducting semi-structured interviewing with recordings and transcripts, the Engineer might simply make their own hand-written notes during their discussions, which they subsequently mark-up, and from which the Bayesian Network policy is learned. However, as is clear from Sect. 3, the validity of the requirements that are elicited very much depend on the skills of the interviewer, and one must be careful not to sacrifice completeness for the sake of expediency.

Acknowledgements. Thanks to Simon O'Donovan who prototyped the Android photograph sharing assistant for his UCC Bachelor's degree project. This work was supported, in part, by Science Foundation Ireland grant SFI/12/RC/2289 and by the Cyber CNI Chair of Institute Mines-Télécom which is held by IMT Atlantique and supported by Airbus Defence and Space, Amossys, EDF, Orange, La Poste, Nokia, Société Générale and the Regional Council of Brittany; it has been acknowledged by the French Centre of Excellence in Cybersecurity.

A Sample policy

A.1 Marked up interview text

```
%A small fragment of the original marked-up interview.
%
\begin{interview}{Bob}{2}
[...]
\qqquestion
do you think if you were out on the street and there was a homeless
person, maybe asleep, and if you looked at that person and thought, I
want to take a photograph of that because I want to make a point about
it, I want to use it sometime, do you think you'd take that kind of
photograph

\answer
no, I wouldn't, I think it's just probably because,
\qaCode{vulnerable}{I think to me it would be exploiting that person
really and considering their circumstances, it's almost like you're
taking, sort of dehumanising that person, almost objectifying them
sort of, so in a sense you're homeless, you're on the street }, so
therefore , \qaDep{suffering}{vulnerable}
\qaCode{(vulnerable)+!share}{I can take a photograph and the intention
might be to publicise the issue, but in actual fact in doing that
you're also sort of putting a negative spin on it as well},
```

```

\qaDep{child}{vulnerable} \qaCode{((child, suffering)+vulnerable)}{but
I feel the same way about, for instance, pictures taken in developing
countries of starving children and stuff like that, and tee shirts
with logos from different organisations because you might argue on one
hand that you have to advertise for the cause and it's important to
raise peoples' consciousness about issues like that, but in saying
that there's also something that I find a bit disturbing in that}
\qaCode{!share+(suffering, child)}{because there's a line that you
don't cross when it comes to} \qaCode{vulnerable+!share}{protecting
people's dignity and privacy} and things like that and I think the
difficulty sometimes is trying to weigh that up.
\qaDep{family}{share} \qaCode{family+share,!family+!share}{It's
probably more straight forward when it's family and friends but if
you're using it in a different context, to serve a different purpose,
I think then maybe it's a bit harder to weigh it up, } [...]
\qaDep{vulnerable}{share}
\end{interview}

```

A.2 Generated Bayesian Network Policy

```

node child
{
  states = ("child" "XXXNOTchild" );
  diagnosistype = "AUXILIARY";
  DSLxSUBMODEL = "Root Submodel";
  ismapvariable = "false";
  ID = "child";
  label = "child";
  DSLxEXTRA_DEFINITIONxDIAGNOSIS_TYPE = "AUXILIARY";
  excludepolicy = "include whole CPT";
}
node vulnerable
{
  states = ("vulnerable" "XXXNOTvulnerable" );
  diagnosistype = "AUXILIARY";
  DSLxSUBMODEL = "Root Submodel";
  ismapvariable = "false";
  ID = "vulnerable";
  label = "vulnerable";
  DSLxEXTRA_DEFINITIONxDIAGNOSIS_TYPE = "AUXILIARY";
  excludepolicy = "include whole CPT";
}
node suffering
{

```



```

states = ("suffering" "XXXNOTsuffering" );
diagnostistype = "AUXILIARY";
DSLxSUBMODEL = "Root Submodel";
ismapvariable = "false";
ID = "suffering";
label = "suffering";
DSLxEXTRA_DEFINITIONxDIAGNOSIS_TYPE = "AUXILIARY";
excludepolicy = "include whole CPT";
}
node family
{
states = ("family" "XXXNOTfamily" );
diagnostistype = "AUXILIARY";
DSLxSUBMODEL = "Root Submodel";
ismapvariable = "false";
ID = "family";
label = "family";
DSLxEXTRA_DEFINITIONxDIAGNOSIS_TYPE = "AUXILIARY";
excludepolicy = "include whole CPT";}
node share
{
states = ("share" "XXXNOTshare" );
diagnostistype = "AUXILIARY";
DSLxSUBMODEL = "Root Submodel";
ismapvariable = "false";
ID = "share";
label = "share";
DSLxEXTRA_DEFINITIONxDIAGNOSIS_TYPE = "AUXILIARY";
excludepolicy = "include whole CPT";
}
potential ( child | )
{
data = ( 0.8553 0.14467 );
}
potential ( vulnerable | suffering child )
{
data = ((( 0.8527 0.1473 )
( 0.8586 0.1414 ))
(( 0.8655 0.1345 )
( 0.8789 0.1211 )));
}
potential ( suffering | )
{
data = ( 0.8481 0.1519 );
}

```

```

potential ( family | )
{
  data = ( 0.2545 0.7455 );
}
potential ( share | family vulnerable )
{
  data = ((( 0.8148 0.1852 )
            ( 1.0 0.0 ))
          (( 0.0964 0.9036 )
            ( 0.0 1.0 )));
}

```

The above Bayesian network, in Hugin .net format, was generated by SamIam [11] using EM-learning on the dataset given in Fig. 5. Note that in this implemented policy, each complementary state ! v is encoded as literal XXXNOT v .

References

1. Adams, A., Lunt, P., Cairns, P.: A qualitative approach to HCI research. In: Cairns, P., Cox, A. (eds.) *Research Methods for Human-Computer Interaction*. Cambridge University Press (2008)
2. Adams, A., Sasse, M.: Users are not the enemy. *CACM* **42**(12), 40–46 (1999)
3. Ahern, S., Eckles, D., Good, N.S., King, S., Naaman, M.: Over-exposed? Privacy patterns and considerations in online and mobile photo sharing. In: *SIGCHI Conference on Human Factors in Computing Systems*, pp. 357–366 (2007)
4. Basin, D., Doser, J., Lodderstedt, T.: Model driven security for process-oriented systems. In: *Symposium on Access control Models and Technologies* (2003)
5. Bellotti, V., Sellen, A.: Design for privacy in ubiquitous computing environments. In: de Michelis, G., Simone, C., Schmidt, K. (eds.) *European Conference on Computer Supported Cooperative Work*, pp. 77–92. Springer, Dordrecht (1993). https://doi.org/10.1007/978-94-011-2094-4_6
6. Breaux, T., Antón, A.: Analyzing regulatory rules for privacy and security requirements. *IEEE Trans. Softw. Eng.* **34**(1), 5–20 (2008)
7. Cadiz, J., Gupta, A.: Privacy interfaces for collaboration. Technical report MSR-TR-2001-82, Microsoft Research, Redmond, WA (2001)
8. Caputo, D.D., Pfleeger, S.L., Sasse, M.A., Ammann, P., Offutt, J., Deng, L.: Barriers to usable security? Three organizational case studies. *IEEE Secur. Priv.* **14**(5), 22–32 (2016). <https://doi.org/10.1109/MSP.2016.95>
9. Charmaz, K.: *Constructing Grounded Theory*. Sage Publications, London (2006)
10. Charmaz, K.: Disclosing illness and disability in the workplace. *J. Int. Educ. Bus.* **3**(1/2), 6–19 (2010)
11. Darwiche, A., et al.: SamIam: Sensitivity Analysis, Modeling, Inference and More. UCLA Automated Reasoning Group. <http://reasoning.cs.ucla.edu/samiam/>. Accessed 07 July 2017
12. Dodier-Lazaro, S., Abu-Salma, R., Becker, I., Sasse, M.A.: From paternalistic to user-centred security: putting users first with value-sensitive design. In: *Proceedings of the 3rd CHI Workshop on Values in Computing* (2017)

13. Dourish, P., Grinter, E., de la Flor, J.D., Joseph, M.: Security in the wild: user strategies for managing security as an everyday, practical problem. *Pers. Ubiquit. Comput.* **8**(6), 391–401 (2004)
14. Firesmith, D.: Security use cases. *J. Object Technol.* **2**(3), 53–64 (2003)
15. Flechais, I., Mascolo, C., Sasse, M.: Integrating security and usability into the requirements and design process. *Int. J. Electron. Secur. Digit. Forensic* **1**(1), 12–26 (2007)
16. Foley, S.N., Rooney, V.M.: Qualitative analysis for trust management. In: Christianson, B., Malcolm, J.A., Matyáš, V., Roe, M. (eds.) *Security Protocols 2009*. LNCS, vol. 7028, pp. 298–307. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36213-2_33
17. Hakkila, J., Chatfield, C.: It’s like if you opened someone else’s letter: user perceived privacy and social practices with SMS communication. In: CHI 05: MobileCHI, 7th International Conference on Human Computer Interaction with Mobile Devices and Services, pp. 357–366 (2005)
18. Inglesant, P., Sasse, A., Chadwick, D., Shi, L.: Expressions of expertness: the virtuous circle of natural language for access control policy specification. In: Symposium on Usable Privacy and Security (SOUPS) 2008, Pittsburg, PA, USA (2008)
19. Jendricke, U., Gerd tom Markotten, D.: Usability meets security - the identity-manager as your personal security assistant for the internet. In: 16th Annual Computer Security Applications Conference (2000)
20. Kvale, S., Brinkmann, S.: *InterViews. Learning the Craft of Qualitative Research Interviewing*, 2nd edn. Sage Publications, London (2009)
21. Lauritzen, S.: The EM algorithm for graphical association models with missing data. *Comput. Stat. Data Anal.* **19**, 191–201 (1995)
22. Massacci, F., Mylopoulos, J., Zannone, N.: Security requirements engineering: the SI* modeling language and the secure tropos methodology. In: Ras, Z.W., Tsay, L.S. (eds.) *Advances in Intelligent Information Systems. Studies in Computational Intelligence*, vol. 265, pp. 147–174. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-05183-8_6
23. Mouratidis, H., Giorgini, P.: Secure tropos: a security-oriented extension of the tropos methodology. *Int. J. Softw. Eng. Knowl. Eng.* **17**(2), 285–309 (2007)
24. O’Connell, D.C., Kowal, S.: Basic principles of transcription. In: Smith, J.A., Harre, R., Van Langenhove, L. (eds.) *Rethinking Methods in Psychology. Part II, Discourse as Topic*, Chap. 7. Sage Publications, London (1995)
25. Onabajo, A., Jahnke, J.: Properties of confidentiality requirements. In: 19th IEEE Symposium on Computer-Based Medical Systems (2006)
26. Parkkola, H., Saariluoma, P., Berki, E.: Action-oriented classification of families’ information and communication actions: exploring mothers’ viewpoints. *Behaviour and Information Technology* **28**(6), 525–536 (2009)
27. Rashid, A., et al.: Discovering “unknown known” security requirements. In: International Conference on Software Engineering. ACM Press (2016)
28. Seaman, C.: Qualitative methods in empirical studies of software engineering. *IEEE Trans. Softw. Eng.* **25**(4), 557–572 (1999)
29. Srivastava, S.: Mobile phones and the evolution of social behaviour. *Behav. Inf. Technol.* **24**(2), 111–129 (2005)
30. Thomas, K., Bandara, A., Price, B., Nuseibeh, B.: Distilling privacy requirements for mobile applications. In: 36th International Conference on Software Engineering (ICSE2014), 31 May–7 June, 2014, Hyderabad, India, pp. 871–882 (2014)
31. Twining, P., et al.: Some guidance on conducting and reporting qualitative studies. *Comput. Educ.* **106**, A1–A9 (2017)

32. Wang, Y., et al.: I regretted the minute I pressed share: a qualitative study of regrets on Facebook. In: 2011 Symposium on Usable Privacy and Security (SOUPS), Pittsburg, PA, USA (2011)
33. Zurko, M.E., Simon, R.T.: User-centered security. In: 1996 Workshop on New Security Paradigms, NSPW 1996, pp. 27–33. ACM (1996)

An Anti-pattern for Misuse Cases

Mohammad Torabi Dashti¹ and Saša Radomirović²(✉)

¹ Department of Computer Science, ETH Zurich, Zurich, Switzerland
mohammad.torabi@inf.ethz.ch

² School of Science and Engineering, University of Dundee, Dundee, UK
s.radomirovic@dundee.ac.uk

Abstract. Misuse case analysis is a method for the elicitation, documentation, and communication of security requirements. It builds upon the well-established use case analysis method and is one of the few existing techniques dedicated to security requirements engineering. We present an anti-pattern for applying misuse cases, dubbed “orphan misuses.” Orphan misuse cases by and large ignore the system at hand, thus providing little insight into its security. Common symptoms include implementation-dependent threats and overly general, vacuous mitigations. We illustrate orphan misuse cases through examples, explain their negative consequences in detail, and give guidelines for avoiding them.

1 Introduction

Misuse case analysis is a method for helping requirements engineers with the notorious task of eliciting security requirements. The elicited requirements are documented textually, or as UML-inspired diagrams, to facilitate communication among business analysts, developers, project managers, and other stakeholders. Similarly to use cases, misuse cases can also form a basis for estimating project cost and efforts. Other applications of misuse cases include documenting the provenance of security functionalities and enabling security testing and risk analysis; see, for example, [16, 17].

To carry out misuse case analysis, first, a system’s functional requirements are elicited as a set of use cases. This step follows the well-established use case analysis method, extensively studied and applied in software engineering. Then, engineers consider each elicited functional use case, and investigate how an adversary might “misuse” it. What constitutes a misuse is determined by the security objectives that are, implicitly or explicitly, available to the engineers.

Finally, to mitigate the misuse cases obtained in the second step, new functional use cases are elicited. These are called security use cases. Optionally, the analysis loops back to the second step for considering threats against the newly added (security) use cases. Functional use cases, security and otherwise, are refined and implemented, whereas misuse cases are fictitious. We assume that the reader has a rudimentary understanding of use cases and misuse cases. For a detailed introduction to these methods see, for example, [1, 16].

Given a problem, an anti-pattern illustrates a recurring solution that has undesired consequences [7]. We present the *orphan misuses anti-pattern*, an anti-pattern for applying misuse case analysis. Intuitively, orphan misuse cases are “orphans” as they ignore the use cases elicited for the system at hand. Consequently, they provide little insight into the system’s security: either the analysis prematurely ends with high-level objectives, or a number of well-known, code-level attacks are selected and listed. In both cases, the system at hand remains by and large unanalyzed. Common symptoms of orphan misuse cases include implementation-dependent threats, such as “buffer overflow,” and mitigations, such as “prevent fraud,” that pertain to (almost) any system and are hence vacuous.

Contributions. We define orphan misuse cases and explain their negative consequences in detail (Sect. 2). Afterward, we present the orphan misuses anti-pattern (Sect. 3) and illustrate it through examples (Sect. 4). We discuss how to avoid orphan misuse cases and give recommendations for writing effective misuse cases (Sect. 5).

2 Orphan Misuse Cases

Any attack can be trivially represented as a misuse case: draw a circle, write the name of the attack inside it, and call it a misuse case. Any preventive or prohibitive mitigation mechanism can similarly be represented as a security use case. This trivial expressiveness comes at a cost: requirements engineers, engaged in misuse case analysis, are left with no guidelines. They are supposed to imagine all possible attacks. Consequently, analysis paralysis may ensue: when imagination is set loose, the outcome is paralyzed because of the undue amount of time and energy that is spent on analysis.

Eliciting security requirements needs a cognitive catalyst: a starting point for thinking about possible attacks. In misuse case analysis, the starting point is a (functional) use case that is already elicited through use case analysis. Focusing on elicited use cases guides the thoughts and limits the search space hence discouraging analysis paralysis. We illustrate this point with a simple example.

Example 1. A “register new clients” use case has been elicited for a web site, associated to a political party, see Fig. 1. The use case includes a mechanism for preventing two clients from having the same email address: trying to register using an email address that exists in the system leads to an error, signaling that the address cannot be reused.

Any number of attacks are imaginable on this system, ranging from command injection to kidnapping system administrators and coercing them into collaboration. The details are also as varied as the imagination permits.

But, focusing only on the logic of this use case brings to light the fact that an adversary can find out, within a margin of error, whether a certain person has an account on the system. Suppose the adversary tries to register using an

email address that belongs to X, a public figure, and receives an error message, indicating that the address cannot be reused. Then, the adversary can infer that X has an account on the system and is likely sympathetic to the party. This violates the (implicit) security objective that a person’s membership in the web site, which betrays a certain political inclination, must not be revealed through the system. This misuse case is denoted “reveal membership” in Fig. 1. \triangle

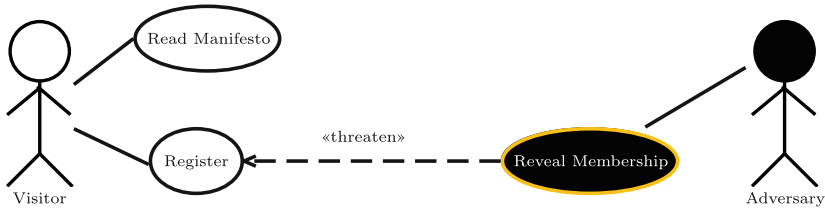


Fig. 1. Analysis for Example 1, drawn with SeaMonster Security Modeling Software

There is a difference between sabotaging a use case and misusing it. For example, command injection attacks hardly constitute a misuse case in the above scenario, because of two reasons. First, a large number of use cases can be attacked through command injection, if their implementation is not based on safe languages. In this sense, command injection has little to do with the specific use case of registering new clients. Therefore, if command injection attacks are to be considered as misuse cases, a myriad of other code-level attacks, including more exotic ones like “row hammering,” [6] should also be included, making the elicitation process practically intractable. We return to code-level threats and their mitigation techniques in Sect. 5.

The second reason is that command injection is not inherently part of, say, a pen-and-paper procedure. A stored procedure on a SQL database might however suffer from it. While the requirements should not assume a particular implementation, command injection presumes a fixed (sort of) implementation. A similar argument shows that kidnapping system administrators is also hardly a misuse of the “register new clients” use case. Considering this type of attack does not provide insight into a problem that the application’s architecture suffers from. In contrast, the “reveal membership” misuse case is inherent to the registration use case with the duplicate detection mechanism described in Example 1. It is a problem that demonstrates how the elicited functionalities can be misused and thus provides insight into a problem that exists at this level of abstraction.

In UML nomenclature, a misuse case amounts to an undesirable use case that “includes” a desired use case, where to include a use case roughly means using it as a subroutine; see [9]. This conforms to the original rationale of Sindre and Opdahl: “many threats to a system can largely be achieved by using that system’s normal functionality” [15]. We call a misuse case that does not “include” one or more functional use cases an **orphan** misuse case. Command injection

and kidnapping system administrators in the above scenario are examples of orphan misuse cases.

We conclude this section with a side note. Security objectives are not elicited through misuse cases. They must be present, implicitly or explicitly, to determine what a misuse is. Therefore, they are not the (main) outcome of a misuse case analysis. What is elicited through misuse cases is the security functions a system must have; see, e.g., [2]. These functions, i.e. security use cases, in effect constrain the system's other use cases. Referring to these as "security requirements" might be slightly misleading, but it is well-accepted in the literature. For example, Haley, Laney, Moffett, and Nuseibeh define a security requirement as a constraint on system functions [5].

3 Orphan Misuses Anti-pattern

We argue against writing orphan misuse cases.

Orphan misuse cases are not tied to the functional use cases that have been elicited for the system at hand. Therefore, they tend to be either overly specific, pertaining to a fixed implementation technology, or overly general, bordering on high-level objectives rather than functions. We illustrate these points through examples from the literature.

Implementation-dependent orphan misuse cases. Sindre and Opdahl give "get privileges" as a misuse case for "register customer" use case [16]. Peterson and Steven give "inject commands" as a misuse case for "review account" in a banking system [12]. Rostad gives "overflow attack" as a misuse for "enter user name" use case, which is mitigated by "input validation" [14].

These examples mix up the abstraction levels: a particular implementation technology is presumed at the requirements elicitation phase. Moreover, "eliciting" code-level orphan misuse cases is a rather futile exercise: they appear to be *recalled* from a list of generic attacks, e.g. OWASP's top ten or CWE's top twenty five, rather than being actually *elicited* for the system at hand. This is not unexpected: in the requirements elicitation phase, often no implementation is available. Therefore, the attacks cannot be about an implementation's peculiarities. This signals the occurrence of analysis paralysis: the engineers, not knowing what to think of, fall back on the well-known attacks, ignoring the elicited use cases.

Note that we do not argue against considering wide-spread, code-level attacks. Rather, the point is that when misuse cases for a given system are elicited, there is hardly any value in thinking about generic well-known attacks. By definition, they are all known, and listed elsewhere. We argue that by shifting the focus towards the use cases at hand, genuine misuse cases can be elicited and specific mitigation mechanisms can be devised and documented as security use cases; see Sect. 4. Moreover, these steps are guided by elicited use cases, which help with analysis paralysis.

Overly general orphan misuse cases. Regev, Alexander, and Wegmann give “launder money” as a misuse case for “establish reputation,” which is mitigated by “check money laundering,” in banking [13]. Pauli and Xu give “impersonate user” as a misuse case for “enter appointment,” which is mitigated by “recognize user,” in a health-care system [11]. Lehtonen, Michahelles, Fleisch give “theft from internal IT” as a misuse case for “tag authentication,” which is mitigated by “secure internal IT system,” in the RFID context [8].

In the examples above, the analysis has terminated prematurely. For instance, “check money laundering” is a high-level objective, as opposed to a functional (security) use case. Overly general orphan misuse cases, and their mitigation, by and large ignore the specific use case under study. We of course do not dismiss the value of high-level objectives. The problem lies with premature termination of analysis: these objectives need further refinements through settling a range of issues regarding adversarial capabilities and the value of the protected assets. We return to this point in Sect. 4.

Table 1. Orphan misuses anti-pattern

Problem. Elicitation of security requirements through misuse case analysis.

Recurrent Solution. Orphan misuse cases which do not include any use case.

The Solution’s Negative Consequences. Orphan misuse cases likely ignore the use cases elicited for the system at hand. They often hide the underlying trade-offs between security and cost, efficiency, usability, and so forth, and provide little insight: either the analysis ends prematurely at the objectives level, or a number of well-known code-level attacks are selected and listed. In both cases, the system at hand is by and large ignored.

Common Symptoms. Implementation-dependent misuse cases, such as “buffer overflow.” Misuse cases that categorically threaten (almost) any use case. Mitigations amounting to vacuous objectives, of the form “secure it!”

How to Avoid It. Follow the logic of use cases and think about how they, as given, can be misused. Let security objectives inform the elicitation process.

We define the **orphan misuses anti-pattern** in Table 1. Next, we illustrate how orphan misuse cases can be avoided.

4 Analysis Without Orphan Misuse Cases

Below, we start with an elicited use case and work out how it can be misused. Mitigating the elicited threats tends to be more complex than recalling well-known security solutions. It often raises fundamental questions about the presumed adversarial capabilities, the expected level of security and its cost. These issues are inherent to security engineering, and in practice they cannot be resolved by requirements engineers alone. Communication among engineers and

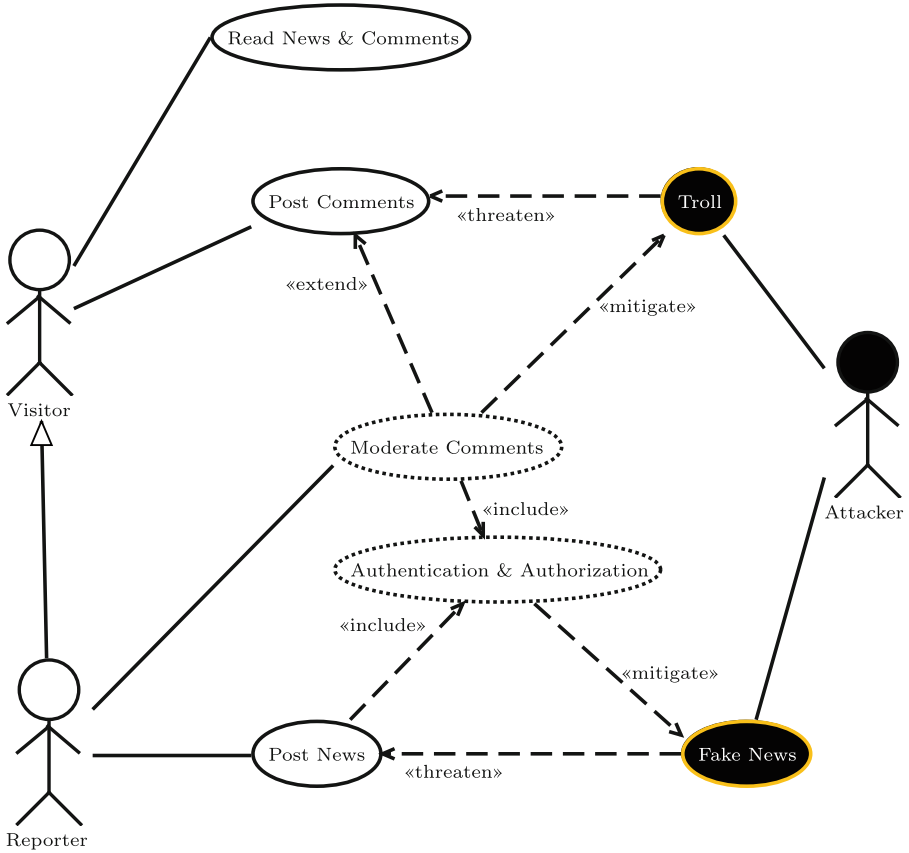


Fig. 2. Analysis for Example 2

other stake-holders is necessary. Compromises are often inevitable, but informed discussions shed light on what is lost for which gain.

Example 2. A news service has two types of users: reporters and visitors. Visitors may read the news, post comments on the news, and read the comments. Reporters inherit visitors’ rights and may also post news items. Obviously, to mitigate fake news, the “post news” use case must include an authentication (security) use case, see Fig. 2.

Rather than thinking about arbitrary attacks that might threaten the system, we focus on the logic of the elicited use cases. We identify “troll” as a misuse of the “post comments” use case. Trolling violates the (implicit security) objective stating that the service’s users must be able to share their opinions safely.

Suppose we identify two possible mitigations: (1) users must authenticate to post a comment. (2) Comments are moderated by reporters before becoming public. The first mitigation is prohibitive: a troll can eventually be identified and

banned. Of course, it is possible that trolls create new accounts to escape the ban, and also it is not clear how a set of comments is decided to be inappropriate.

The second mitigation, which is preventive, may unduly increase the reporters' work load. Moreover, it enables censorship: rogue reporters can misuse the "moderate comments" use case to silence the commentators who challenge their views. This violates the security objective above. To mitigate the "censor" misuse case, a separation of powers mechanism can prevent reporters from moderating comments on their own posts. A preliminary analysis, omitting the censor misuse case, is shown in Fig. 2.

Which mitigation should be adopted? This question cannot be answered by requirements engineers alone. The options, including the cost of enforcing separation of powers, must be communicated to the stake-holders. This facilitates informed discussions for deciding a specific mitigation. \triangle

Misuse case elicitation in Example 2 is focused on the news service, but it is not about well-known, implementation-dependent attacks. These attacks are important issues in practice, but not in the requirements elicitation phase. Moreover, until we get the requirements right, and architect the system accordingly, there is little hope for securing the system even if all code-level attacks are accounted for.

Example 3. In a perimeter control system, the "unlock all doors" use case has been elicited following fire safety regulations: in case of fire all doors must be unlocked. Obviously, this use case should not be publicly accessible. An authentication and authorization (security) use case is due: to activate the use case, one must have a certain role in the organization. This step is rather obvious. The interesting question is what to do about sensitive areas that are left unprotected in case of fire.

Suppose there is a safe in the building. An arsonist adversary might start a fire to ease his/her access to the safe. Suppose we identify two mitigations: (1) a surveillance system, resilient to fire, monitors the safe and the paths that lead to it. (2) In case of fire, aqueous foam is dispensed, which quickly expands and fills the area around the safe, hence delaying the adversary's access [4].

The first mitigation is prohibitive and the second one preventive, but it comes with high maintenance costs. Which one should we choose? Again, the decision is not for requirements engineers to make. The presumed capabilities of the adversary against whom the perimeter is protected, the value of the asset, i.e. the safe, and technological constraints, e.g. the availability and dependability of foam dispensing equipments and their suppliers, are among the questions that must be answered before choosing a concrete mitigation.

Continuing with the misuse case analysis reveals that the first mitigation is privacy-intrusive, e.g. surveillance cameras can be misused by an insider to track a person's movements. These findings help the stake-holders to make informed decisions. \triangle

In Example 3, it is necessary to make risks observable through, e.g., clarifying adversarial capabilities, and evaluating the worth of protected assets. This is because the focus is on a concrete system. In contrast, an orphan misuse case, which is divorced from the system, does not raise these issues naturally. Writing high-level objectives such as “mitigate arson” is trivial, but without addressing the above issues regarding adversaries, assets, and so forth, this mitigation is vacuous, hence of little use to the developers and other stake-holders.

Example 1’s misuse case can be subjected to a similar analysis for finding suitable mitigations. For instance, strengthening the error handling mechanism might appear promising. We do this next.

Example 4 (Continuing Example 1). Consider the register new clients use case. We need a mechanism that prevents two clients from registering the same email address. As discussed, if the mechanism displays an error message on the same channel used to enter email addresses, then an adversary can infer that the owner of an email address has an account on the system. Suppose we change the mechanism to include a confirmation loop, as explained below.

1. When an email address is provided in the registration step, a message is sent to that email address, its contents determined as follows.
 - 2a. If the email address is already registered in the system the recipient is informed that someone tried to register with this address. The recipient is thus reminded that they do have an account.
 - 2b. If the email address is not registered, a link is provided to continue with the registration.

This prevents the “reveal membership” misuse case as the adversary cannot differentiate between the two types of email that are sent to an address that he does not own.

Further extending the analysis can now produce a new misuse case where an attacker is able to misuse the confirmation loop to send unsolicited emails (spam) to arbitrary email addresses. To combat spamming, the number of emails sent to any given email address must be controlled. Clearly, this mitigation comes at the cost of a more complex, stateful error handling mechanism. The resulting diagram is shown in Fig. 3. △

We conclude this section by remarking that examples where the elicitation of misuse cases follows the available use cases as a guideline are also found in the literature. The information misuse case given in Example 1 is based on an example of Sindre and Opdahl [16]. OWASP’s Testing Guide gives “brute force,” i.e. repetitive password guessing, as a misuse of the “authentication” use case, which is mitigated by various checks added to the use case [10]. This too follows the guideline.

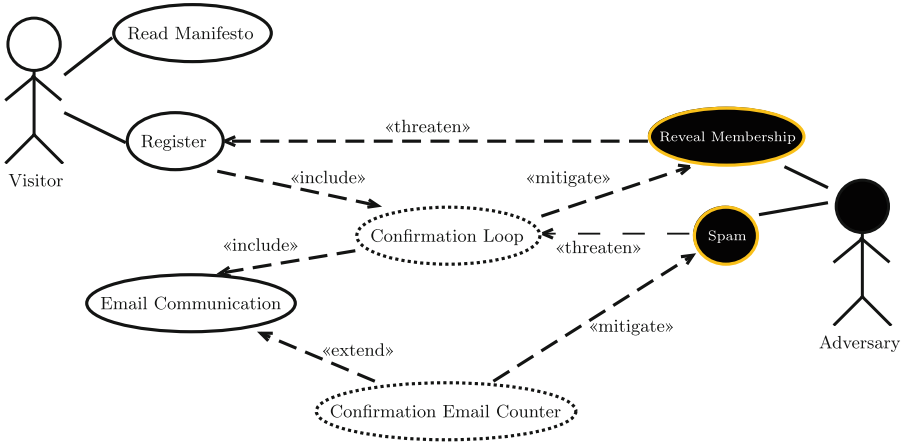


Fig. 3. Analysis for Example 4

5 How to Avoid Orphan Misuse Cases

Writing orphan misuse cases is trivial. High-level misuse cases always amount to the same thing: “sabotage it!” Code-level misuse cases are also not hard to come by. Looking up any of the existing top ten or top twenty five vulnerability lists is a good start. We dismiss such misuse cases as products of an anti-pattern. Since they imitate the elicitation process but do not contribute to it, the system at hand remains unanalyzed.

We advocate writing misuse cases that are specific to the system under study. This is hard: it demands new thinking and fresh perspectives for each case study. The result is however of higher quality as it is coupled with the system at hand, and moreover the process encourages thinking about fundamental issues, such as the presumed adversarial capabilities and cost-benefit analysis.

Clearly avoiding orphan misuse cases means that not all attacks can be accounted for in misuse case analyses. This is a limitation, but, we argue, a desirable one:

The point of misuse cases is to elicit the security functions that a system must have. Therefore, a successful analysis, no matter how unattainable, is one that covers all necessary security functions for a given system, in light of the adversarial capabilities, cost-benefit analysis, and other forms of compromise inherent to usable, practical security. These security functions, similarly to other functional use cases, are likely to be flawed when implemented. But, there is little we can do about it at the requirements elicitation phase: it is unreasonable to expect misuse cases to account for all security issues. Combating code-level vulnerabilities that could lead, for instance, to malicious code-injection has its own dedicated techniques, such as testing and code inspection, which fall outside the domain of requirements engineering per se.

Pushing every concern to the requirements elicitation phase is impractical for at least two reasons. First, analysis paralysis is real: the list of attacks is virtually inexhaustible. A comparison to chess is instructive here. Once in a while a grand master comes up with a brilliant new attack (or defense) strategy that has eluded thousands of people who played and studied chess for centuries. Then, it should not be surprising that we cannot foresee all attacks against industrial-scale computer systems: these systems are substantially more complex and more opaque than chess.

Second, and more importantly for our argument, genuine security issues, with architectural implications, can be discovered and mitigated at the requirements level, only if we shift the focus towards them. We claim that such issues can be discovered more effectively, and with less cluttering, when orphan misuse cases are avoided. We briefly illustrate this on the application programming interfaces (APIs) that a mobile operating system exposes to application developers. Each API corresponds to a functionality. For example, the API for microphone access enables an application to record sound. This functionality can be misused to undermine privacy objectives and this misuse case can be mitigated by a functionality for the user to disable an application's access to microphone. The sheer number of APIs that an operating system exposes makes the security analysis at this level alone very complex. Security issues due to API misuse, uncovered for example in [3], support this claim.

We now summarize our discussions with five simple, rule-of-thumb guidelines for avoiding orphan misuse cases.

Think of include. A misuse case “threatens” a use case, in the same way one use case “includes” another one. Revise the misuse cases that do not include any use cases and those that categorically “threaten” (almost) any use case.

Go bottom-up. Start with a use case and work out its misuses. Do not start with an adversary with the intention of brainstorming the ways it can attack the system. There are too many ways, hence leading to analysis paralysis.

Respect the abstraction level. If functional use cases are at the requirements level, misuse cases should not belong to the objectives level, nor should they encroach on implementation details.

Make risks observable. If issues such as adversarial capabilities, value of the protected assets, and cost-benefit compromises do not cross your mind, you are likely ignoring the system at hand. Similarly, decisions that can be made without consulting stake-holders are likely the trivial ones. Make risks observable by explicating underlying assumptions and trade-offs.

Dismiss trivial expressiveness. Anything can be written as a misuse case. However, requirements representation is not the same as requirements elicitation. As a representation tool, misuse cases can document any requirement. But, as an elicitation tool, misuse case analysis must be guided by use cases.

We conclude the paper with a note on empirical validation. To empirically study the value of the rule-of-thumb guidelines and the orphan misuses anti-pattern, one must raise one's sights to look beyond symptoms and target root causes as well. Namely, for each flaw found in a system, one must look beyond its

immediate source, and identify the reason, i.e. deficiencies in the requirements engineering phase (and other system development phases) that have led to the flaw. The value of this form of root-cause analysis is well-known; see, for example, Van Vleck's *three questions* [18].

An industrial-scale root-cause analysis, for the purpose of evaluating our requirements elicitation approach and the misuse case analysis method in general, demands substantial efforts and is left for future work. An observation we have made in our lectures is nonetheless illustrative: students stop throwing arbitrary attack names, after we present them with the constraint that a misuse case must include a functional use case as given. Reciting the latest hacker news item becomes irrelevant. Creative thinking, guided by use cases, takes its place.

References

1. Cockburn, A.: *Writing Effective Use Cases*. Addison-Wesley, Boston (2001)
2. Firesmith, D.: Security use cases. *J. Object Technol.* **2**(3), 53–64 (2003)
3. Fratantonio, Y., Qian, C., Chung, S.P., Lee, W.: Cloak and dagger: from two permissions to complete control of the UI feedback loop. In: *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22–26, 2017*, pp. 1041–1057. IEEE Computer Society (2017)
4. Garcia, M.L.: *The Design and Evaluation of Physical Protection Systems*. Elsevier Science, Burlington (2001)
5. Haley, C., Laney, R., Moffett, J., Nuseibeh, B.: Security requirements engineering: a framework for representation and analysis. *IEEE Trans. Softw. Eng.* **34**(1), 133–153 (2008)
6. Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J.-H., Lee, D., Wilkerson, C., Lai, K., Mutlu, O.: Flipping bits in memory without accessing them: an experimental study of DRAM disturbance errors. In: *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA*, pp. 361–372. IEEE Computer Society (2014)
7. Koenig, A.: Patterns and antipatterns. *JOOP* **8**(1), 46–48 (1995)
8. Lehtonen, M.O., Michahelles, F., Fleisch, E.: Trust and security in RFID-based product authentication systems. *IEEE Syst. J.* **1**(2), 129–144 (2007)
9. Object Management Group: *Unified modeling language (OMG UML), version 2.5* (2015)
10. OWASP: *Testing guide v. 4*. <https://www.owasp.org>. Accessed Apr 2016
11. Pauli, J.J., Xu, D.: Misuse case-based design and analysis of secure software architecture. In: *Proceedings of the International Conference on Information Technology: Coding and Computing, ITCC 2005, vol. 2*, pp. 398–403. IEEE Computer Society (2005)
12. Peterson, G., Steven, J.: Defining misuse within the development process. *IEEE Secur. Priv.* **4**(6), 81–84 (2006)
13. Regev, G., Alexander, I.F., Wegmann, A.: Modelling the regulative role of business processes with use and misuse cases. *Bus. Process Manage. J.* **11**(6), 695–708 (2005)
14. Rostad, L.: An extended misuse case notation: Including vulnerabilities and the insider threat. In: *Working Conference on Requirements Engineering: Foundation for Software Quality (RREFSQ)*, pp. 33–34. Essener Informatik Beitrage (2006)
15. Sindre, G., Opdahl, A.L.: Eliciting security requirements by misuse cases. In: *Proceedings 37th International Conference on Technology of Object-Oriented Languages and Systems. TOOLS-Pacific 2000*, pp. 120–131 (2000)

16. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. *Requirements Eng.* **10**(1), 34–44 (2005)
17. van Wyk, K.R., McGraw, G.: Bridging the gap between software development and information security. *IEEE Secur. Priv.* **3**(5), 75–79 (2005)
18. Van Vleck, T.: Three questions about each bug you find. *ACM SIGSOFT Softw. Eng. Notes* **14**(5), 62–63 (1989)

Decision-Making in Security Requirements Engineering with Constrained Goal Models

Nikolaos Argyropoulos^(✉), Konstantinos Angelopoulos,
Haralambos Mouratidis, and Andrew Fish

School of Computing, Engineering and Mathematics,
University of Brighton, Brighton, UK

{n.argyropoulos,k.angelopoulos,h.mouratidis,andrew.fish}@brighton.ac.uk

Abstract. Selecting security mechanisms for complex software systems is a cumbersome process. The presence of multiple goals and architectural components, as well as cost and performance considerations, render decision-making a crucial but complicated aspect of a system's design. In our work, we extend Secure Tropos, a security requirements engineering methodology, by introducing the concept of Risk in order to facilitate the elicitation and analysis of security requirements and also support a systematic risk assessment process during the system's design time. Next, we use Constrained Goal Models to reason about optimal security mechanism combinations with respect to multiple objectives of the system-to-be, taking into account conflicting functional and non-functional goals. This type of reasoning allows combining linear multi-objective optimisation with logical constraints introduced by the system's stakeholders. Finally, we illustrate the application of approach through a real-world case study from the e-government sector.

Keywords: Information security · Security requirements
Decision making · Constraint goal models

1 Introduction

The advances of technology have dramatically increased user expectations of modern information systems. The continuous growth of the number of goals these systems are expected to satisfy, as well as the complexity of their architectures, render software (re-)configuration a challenging process. In particular, information systems which are exposed to cyber-threats must be able to respond to continuous changes in their environment that could put their valuable assets at risk.

The selection of appropriate security configurations should take into consideration the threat landscape in which the system will operate. Therefore, the effects of vulnerabilities and threats towards a system's goals and their mitigation by security countermeasures, should play an important role during the system's design process [29]. The ever-changing nature of the threat landscape

is further amplified by new paradigms in information system architecture (e.g., cloud computing, Internet of Things) [13]. In such volatile environments the risk posed by a threat can greatly vary depending on the impacted system component or the likelihood of a vulnerability being exploited. Therefore, a flexible approach towards risk-aware decision-making is crucial during system design, in particular with regards to the system's security countermeasure configuration. Nonetheless, attempts to provide risk-aware decision support should also be able to take into account trade-offs between security and other functional and non-functional system goals. Thus, striking a balance between effective risk management and functional system design can be a challenging endeavour.

To overcome such challenges, in this paper we extend Secure Tropos in order to support risk-aware decision-making for the design of secure system configurations. Towards this direction, risk related concepts and attributes are integrated into Secure Tropos, allowing designers to express the level of security of their systems as cost-functions. Next, we use Analytic Hierarchy Process (AHP) [25] to estimate the likelihood of threats to be manifested. Our approach provides a new framework that selects optimal security configurations with respect to the severity of threats and the priorities of other goals. More specifically, we express the level of mitigation of each threat and other goals of the systems (e.g. cost and performance) as cost-functions that our proposed framework optimises lexicographically i.e. best adaptations are selected relative to the most important cost function and the best among those are selected relative to the second most important cost function, etc.

Next, in Sect. 2 we introduce central concepts of relevant research areas. In Sect. 3 we describe how the Secure Tropos meta-model is extended in order to support the notion of risk, as well as the instantiation of the basic risk assessment variables. In Sect. 4 we illustrate our approach through a case study. Finally, in Sect. 5 we discuss related works and in Sect. 6 we offer conclusions and directions of future work.

2 Research Baseline

In this section we provide a research baseline for the Secure Tropos approach, constrained goal models, and risk management, which are the main building blocks of our proposed approach.

Goal Models. Secure Tropos, and therefore our approach, adopt the principles of Goal Oriented Requirements Engineering (GORE). The centrepiece of GORE is the concept of goal [8] that captures the intentions of stakeholders. Goals are gradually refined to more detailed goals using AND/OR boolean relationships. The refinement process ends when each goal is refined to detailed tasks, named plans in the Secure Tropos terminology, that can be assigned to a human or software component. Goals and plans express only functional requirements of the system therefore, the concept of softgoal [7] is used to express non-functional requirements. In Giorgini *et al.* [12], the fulfilment or not of a goal is characterised

by four propositions (FS, PS, FD, PD), representing full and partial satisfaction or denial respectively. In our work we do not consider partial propositions.

Security Requirements Engineering. Secure Tropos [21] is a security-oriented extension of the Tropos methodology [4], able to support the elicitation, analysis of security requirements from the early stages of the system development life-cycle. It utilises standard goal-oriented requirements engineering concepts (e.g., actors, goals, dependencies) but also introduces concepts from the domain of security engineering (e.g., security constraints, threats, security mechanisms). Secure Tropos facilitates system design and (security) requirements elicitation through a number of interrelated modelling views. The *Security Requirements* view is used to present the goal decomposition of each system actor and the dependencies between them. Additionally, security constraints and threats are identified and connected to goals and resources in the same modelling view, while potential security mechanisms are identified to satisfy the identified constraints and threats. The *Security Attacks* view is an additional diagram, unique for each of threat identified in the Security Requirements view, which further decomposes each threat to identify its attack methods, the system vulnerabilities they exploit and the coverage provided by the proposed security mechanisms against such vulnerabilities. The use of online threat repositories (e.g., CAPEC [19]) and the consultation of security experts are recommended for the identification of threats, attack methods and vulnerabilities and the derivation of sets of potential security mechanisms. A detailed presentation of the components and modelling views of the Secure Tropos methodology is presented in [20].

Other than its support for security modelling in an explicit and structured manner, Secure Tropos has been selected as the basis of our approach due to: (i) its social concepts (e.g., actors, goals, dependencies) and analysis capabilities during the early requirements stage; (ii) the simultaneous consideration of security along with the other requirements of the system-to-be; and (iii) its ability to support the design stage of systems development, through the mapping of abstract security constraints and threats to specific security mechanisms [1].

Constrained Goal Models. Goal models often present high variability, expressed by multiple alternative solutions to fulfil one or more goals. One of the tasks of GORE is to decide which from these alternatives should be implemented in the system-to-be and which should not. Given the nature of goal models, each goal practically represents a predicate that relates with other predicates through AND/OR relationships, construction first order logic formulas. In traditional goal modelling approaches goals are assigned with weights to represent their relative importance for the system and provide the means for comparing alternative solutions. Hence, goal reasoning in such approaches means finding a solution to a maximum satisfiability (MAX-SAT) problem.

As systems became more complex, the selection criteria became more sophisticated. More variables rather than the relative importance are associated with goals. Such variables are usually related to cost, performance, energy consumption etc. This emerged the field of satisfiability and optimisation modulo theories (SMT/OMT), where formulas are also associated with linear equations that must

be optimised by the any solution found for the satisfiability problem. The integration of SMT/OMT with goal models has been implemented by Constrained Goal Models (CGMs) [22]. Such goal models allow the definition of (a) multiple variables associated with the modelled goals and (b) linear equations composed by these variables that should be optimised. Therefore, along with the satisfiability problem that is native to goal models, a multi-objective optimisation problem should be solved in parallel. This is done with the use of a scalable external reasoner, OptiMathSAT [26], which is invoked to find optimal solutions over CGMs.

Risk Management. In the field of information security, a risk expresses the potential of a threat to exploit vulnerabilities of organisational assets and as a result harm the organisation [14]. Risk management is a set of coordinated activities performed by an organisation to minimise the effects of risks [15].

Risk assessment is the initial phase of the risk management process, during which organisations elicit potential threats and the vulnerabilities they exploit to threaten the functionality of their systems. The risk introduced by such vulnerabilities is evaluated and security countermeasures for reducing or eliminating the identified risk are recommended [27]. Values for the impact and likelihood of each identified vulnerability can be estimated using either quantitative or qualitative metrics [3]. The consensus approach for assigning a value to the risk introduced by each vulnerability is by calculating product of its impact and likelihood [23,27]. The overall risk introduced by a threat can then be calculated as the sum of the individual risk values of each of its associated vulnerabilities.

Risk reduction via the use of countermeasures is amongst the most established strategies for risk mitigation. Countermeasures need to be prioritised in terms of the coverage they provide against each risk but also their contribution towards other non-functional objectives of the system (e.g., financial cost, technical constraints, usability) [14]. The risk remaining after the application of a risk mitigating strategy is known as the residual risk. The final phase of the risk management process involves the continuous evaluation and assessment of the implemented system throughout its life-cycle, to account for potential changes to its composition and execution environment.

3 Capturing Risk with Secure Tropos

Secure Tropos introduces a conceptual basis which facilitates security trade-off modelling and analysis [11]. An inherent limitation of all Tropos based approaches is their lack of precise semantics for the quantitative evaluation of system behaviours, including security and risk coverage [5]. Additionally, concepts necessary for the risk analysis process (e.g., risk) are missing. Attempts to align it with risk-related concepts have been developed [17], but they lack the ability to quantitatively perform risk assessment and support a fine-grained security trade-off analysis. To that end, we extend Secure Tropos with a number of concepts and attributes, as presented in Fig. 1 in bold and italic font.

probable is the exploitation of a vulnerability by a certain threat compared to another one. Therefore, likelihood represents a different prioritisation of vulnerabilities with respect to their probability of being exploited and is also estimated using AHP. In contrast to its impact value, which is unique for its vulnerability, the likelihood value depends on the combination of a threat-vulnerability pairing, as the same vulnerability can be exploited by more than one threat but with a different likelihood.

The initial amount of risk introduced by a threat is an aggregation of the risk introduced by each of the vulnerabilities exploited by the threat and is captured by the *InherentRisk* attribute of the *Risk* concept. The amount of risk remaining after risk treatment is applied is captured by the *ResidualRisk* attribute. Additionally, the attribute *ResidualRiskThreshold* captures the maximum accepted amount of residual risk for each threat by the system stakeholders.

The concept of the *Security Mechanism*, which Secure Tropos uses to model technologies utilised to implement the system's security objectives, is extended with a number of attributes. These attributes will allow us to evaluate the contribution of each security mechanism towards the achievement of each of the system's soft-goals (*SoftGoalContribution*) and the mitigation of each identified vulnerability (*VulnerabilityMitigation*).

Finally the *Coverage* attribute has been added to the *Soft Goal* concept to capture the total coverage provided to each by the selected sets of security mechanisms.

3.2 Risk Assessment

The newly introduced concept of Risk and additional attributes to the existing Secure Tropos concepts facilitate the definition of functions which can be used to guide the risk-based adaptation process. More specifically:

Definition 1. Let L , where $L \in \mathbb{R}$ and $0 \leq L \leq 1$, be the Likelihood of a vulnerability to be manifested, I , where $I \in \mathbb{R}$ and $0 \leq I \leq 1$, its Impact, V , where $V \in [0, 1] \subseteq N$, the exploitation of a vulnerability by a threat ($V = 1$) or not ($V = 0$), and n , where $n \in \mathbb{N}$ and $0 \leq N \leq 100$, the total number of identified vulnerabilities in the system. Then, we define as R_I the overall Inherent Risk introduced by a threat, where:

$$R_I = \sum_{i=1}^n (L_i \times I_i \times V_i) \quad (1)$$

Definition 2. Let M , where $M \in \mathbb{R}$ and $0 \leq M \leq 1$, be the Vulnerability Mitigation of a security mechanism towards a vulnerability and m , where $m \in N$, the total number of security mechanisms mitigating that vulnerability. Then, we define as R_M the overall Mitigated Risk of a threat, where:

$$R_M = \sum_{i=1}^n (L_i \times I_i \times V_i \times \sum_{j=1}^m \frac{M_{ji}}{m}) \quad (2)$$

Definition 3. Let R_R be the Residual Risk of a threat, then:

$$R_R = R_I - R_M \stackrel{(1),(2)}{=} \sum_{i=1}^n [(L_i \times I_i \times V_i) \times (1 - \sum_{j=1}^m \frac{M_{ji}}{m})] \quad (3)$$

The process for deciding what mechanisms should be implemented includes four steps:

Step 1: Security Analysis. The system designers along with the security engineers produce the Secure Tropos diagrams that we described in Sect. 2. These models reveal all the Threats to the system's goals and assets and propose alternative solutions, in form of security mechanisms, to mitigate them.

Step 2: Likelihood Estimation. For each vulnerability, estimate a likelihood value using AHP for each threat. When analysing a vulnerability, security engineers assign a likelihood value for each threat that affects this vulnerability.

Step 3: Impact Estimation. For each vulnerability, estimate an impact value using AHP. To elicit such values, security engineers must perform pairwise comparison for all vulnerabilities and prioritise them based on how much the system will be affected if the examined vulnerability is exploited by a threat.

Step 4. Risk Minimisation. Minimise the Residual Risk by using the optimisation functionality of the extended Secure Tropos. This functionality, proposes a set security mechanisms that minimise the Residual Risk taking also into account other goals, such as Cost and Performance.

As we mentioned earlier, new types of attacks are continuously being developed and new vulnerabilities are discovered as software systems evolve. This means that more variables can be introduced to our optimisation problem and the previously estimated values for likelihood and vulnerability, might not be valid anymore. Therefore, in order for a system to remain updated in terms of security, the process above should be repeated periodically.

4 Case Study

For the evaluation of the proposed approach a case study has been performed focusing on an information system for the registration of citizens to a public swimming pool facility at the Municipality of Athens, Greece. A goal model of the system is presented in Fig. 2.

4.1 System Description

The main participants of the system, denoted as actors at the goal model, interact with each other through dependency relationships to achieve their goals. More specifically, the actors involved in this system are the following:

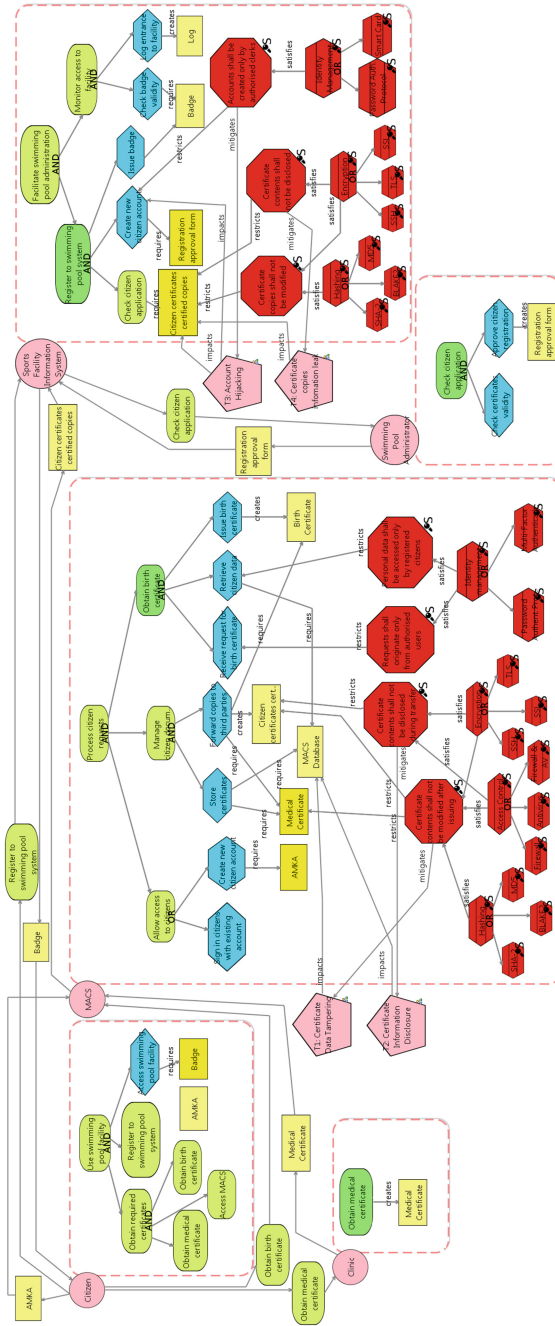


Fig. 2. The Secure Tropos goal model of the Swimming Pool Administration IS

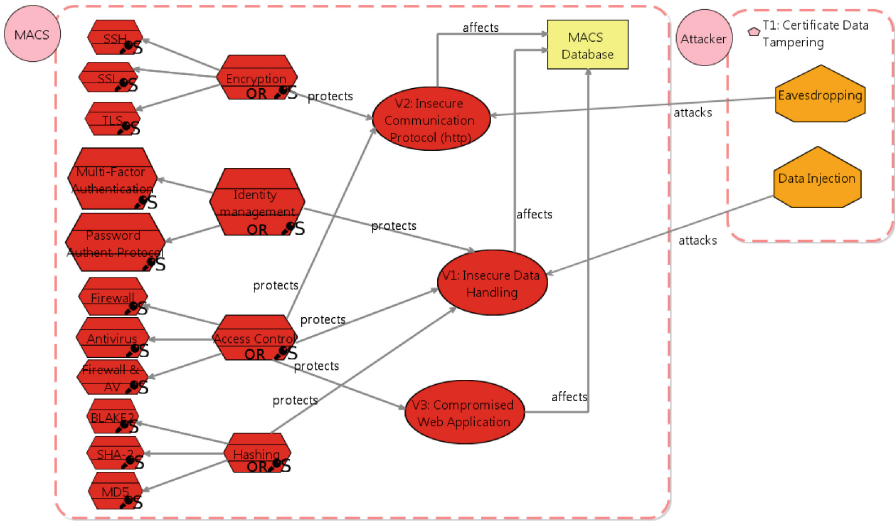
(i) the *Citizen* aiming to register for using the swimming pool facilities, (ii) the medical *Clinic* that examines the citizen and issues a medical certificate, (iii) the *Municipality of Athens Citizen Services (MACS)* system that citizens can use to request and store certificates, (iv) the *Swimming Pool Information System* the gathers copies of the necessary certificates, registers citizens and tracks the usage of the facilities and (v) the *Swimming Pool Administrator* that verifies the validity of the citizen's certificates and approves their registration to the facilities. With the collaboration of the system's designers, the goals of the participating system actors were further decomposed as sub-goals and plans and the documents and infrastructure created and/or utilised throughout the process were captured as resources. Additionally, non-functional goals (soft-goals) that the overall system should satisfy were defined by its stakeholders. More specifically, the first non-functional goal was to keep the implementation costs at a minimum and the second was to maintain a low system complexity in order not to introduce significant overhead in terms of system performance.

4.2 Application

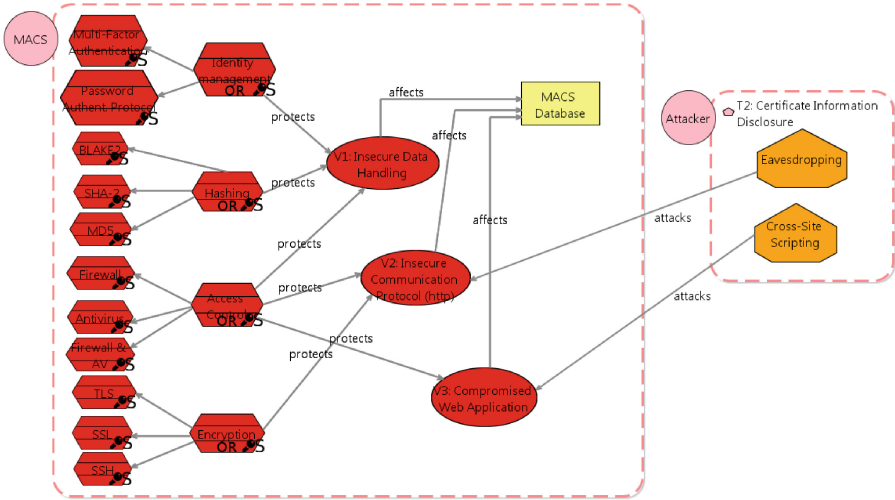
According to Step 1 of the risk management process, as described in the previous section, the security requirements of the system were elicited in the form of security constraints and potential threats along with the vulnerabilities they exploit were identified. Such security constraints formed the basis upon which the security analysis of the system was performed. The security constraints, restricting certain goals or resources of the system, were identified by the system stakeholders and connected to the relevant model elements during the security analysis process. For instance, "*Certificate contents shall not be modified after issuing*", was a constraint identified for the EMACS system and connected to the resources representing the medical and birth certificate at the goal model. Each of the identified security constraints were also assigned to the type of security objective (e.g., authentication, authorisation, confidentiality, integrity, availability) they accomplish.

Through the use of relevant resources (e.g. CAPEC [19]) a number of threats were identified connected to elements of the system they can potentially impact. For instance, the threat of "*Account Hijacking*" was identified for the Swimming Pool Information System which could potentially impact the accomplishment of the "*Create citizen account*" plan and the "*Citizen certificates certified copies*" resource. A further breakdown of the threat manifestation and countermeasures for each of the identified threats is provided by the Security Attacks modelling view, as explained below. Finally, a variety of security mechanisms were proposed in order to both satisfy the system's security objectives and mitigate the identified threats. The mechanisms were grouped according to their functionality, therefore, "*Encryption*", for instance, could be implemented by any of the identified security mechanisms connected to it (i.e., SSH, SSL, TLS).

The Security Attacks view supported by the Secure Tropos approach provides an in-depth view of each of the identified threats and their interaction with the rest of the system. For each threat a number of attack methods are



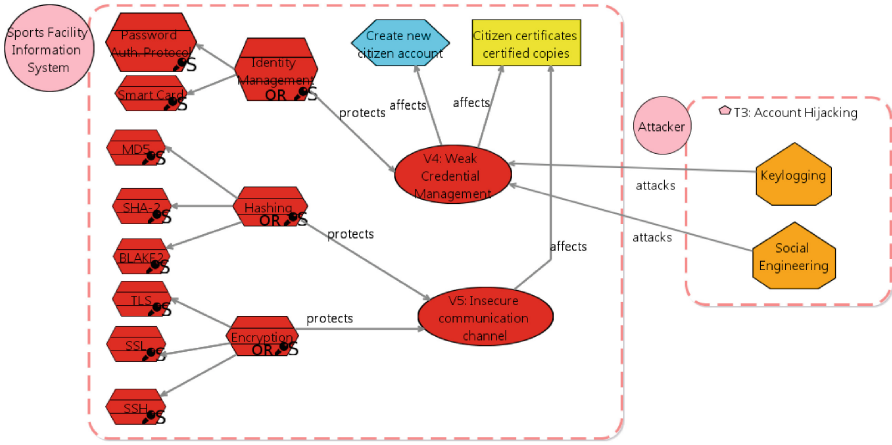
(a) T1: Certificate Data Tampering



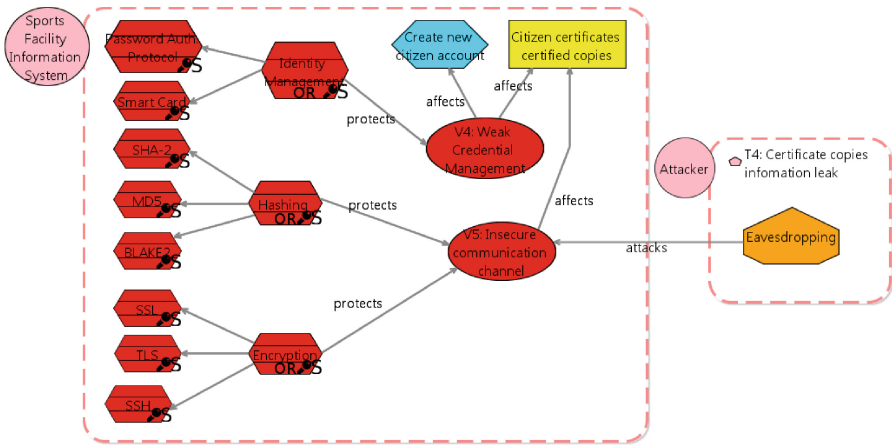
(b) T2: Certificate Information Disclosure

Fig. 3. Security Attacks views of threats T1 and T2

identified, each of which targets one or more vulnerabilities of the system. Such vulnerabilities can be identified both by analysing the system’s architecture and via specialised vulnerability repositories (e.g. CVE database). The same sources can also be used for identifying security mechanisms which can protect the system against such vulnerabilities. The vulnerabilities exploited by each of the identified threats and the types of security mechanisms protecting against each



(a) T3: Account Hijacking



(b) T4: Cert. Copies Information Leak

Fig. 4. Security Attacks views of threats T4 and T5

of those vulnerabilities, as visualised in the Security Attacks views diagrams of Figs. 3 and 4, are summarised in Table 1. By the application of Steps 2 and 3 of the proposed approach, impact and likelihood values were also assigned to the identified vulnerabilities using AHP, as presented in Table 2. Then, for finding a satisfied goal model, for each agent, all the security constraints must be satisfied. Given that a security constraint is satisfied when at least one of the proposed security mechanisms is proposed and a security mechanism might satisfy multiple constraints, the last step of our process selects a set of mechanisms to be implemented that satisfies all the security constraints while minimising risk, as defined in the previous section.

Table 1. Threat - vulnerability - security mechanisms matching

Threat	Vuln.	Encryption	Access control	Hashing	Id Mgmt (EMACS)	Id Mgmt (SP)
T1	V1		✓	✓	✓	
T1, T2	V2	✓	✓			
T2	V3		✓			
T3	V4					✓
T4	V5	✓		✓		

Threats: T1: *Certificate Data Tampering*, T2: *Certificate Information Disclosure*, T3: *Account Hijacking*, T4: *Certificate Copies Information Leak*

Vulnerabilities: V1: *Insecure Data Handling*, V2: *Insecure Communication Protocol*, V3: *Compromised Web Application*, V4: *Weak Credential Management*, V5: *Insecure Communication Channel*

Table 2. Threat - vulnerability value assignment

Threat	Vulnerability	Impact	Likelihood	Inherent risk
T1	V1	0.15	0.4	0.15
	V2	0.15	0.6	
T2	V2	0.15	0.25	0.2625
	V3	0.3	0.75	
T3	V4	0.25	1	0.25
T4	V5	0.15	1	0.15

A number of scenarios have been elaborated and applied to the swimming pool administration system to illustrate the application of our approach. For the identification of the optimal security implementation for each scenario, as dictated by Step 4 of the proposed approach, our system is modelled as a constraint goal model and used as input to the OptiMathSAT solver. The variables used to define each scenario are the following: (i) the residual risk of each threat as a percentage of its initial inherent risk, which is defined as an aggregate of all the vulnerabilities exploited by each threat, (ii) the added cost and (iii) the added performance overhead of the implementation. In each scenario these variables can either have a specific hard threshold or they can be set to be minimised (*min*). Additionally, the minimisation of each variable can be prioritised against the rest of the variables of each scenario. For instance, a scenario may have the minimisation of cost or the reduction of a specific threat as its top optimisation priority. The thresholds and priorities of each variable for each scenario are presented in Table 3.

To obtain values for these variables for each scenario, each of the proposed security mechanisms is instantiated with numerical values regarding the

percentage of mitigation it offers for each of the system’s vulnerabilities and its contribution towards added system cost and performance. An overview of the values assigned to the mechanisms of the swimming pool administration system is provided at Table 4. The resulting security configurations, presented in Table 5, is a combination of the proposed mechanisms which optimally satisfies the parameters of each scenario, as summarised in Table 3.

Table 3. Variable values and thresholds per adaptation scenario

Variable	Scenario					
	1	2	3	4	5	6
T1 Res. Risk ($R_{R(T_1)}$)	<i>min</i>	<i>min</i> ^[1]	<i>min</i> ^[3]	<i>min</i> ^[2]	<25%	<50%
T2 Res. Risk ($R_{R(T_2)}$)	<i>min</i>	<i>min</i> ^[2]	<i>min</i> ^[4]	<i>min</i> ^[3]	<25%	<50%
T3 Res. Risk ($R_{R(T_3)}$)	<i>min</i>	<i>min</i> ^[3]	<i>min</i> ^[5]	<i>min</i> ^[4]	<i>min</i>	<75%
T4 Res. Risk ($R_{R(T_4)}$)	<i>min</i>	<i>min</i> ^[4]	<i>min</i> ^[6]	<i>min</i> ^[5]	<i>min</i>	<50%
Added cost coverage	<i>min</i>	<i>min</i> ^[5]	<i>min</i> ^[1]	<i>min</i> ^[6]	<i>min</i> ^[1]	<i>min</i> ^[1]
Performance overhead coverage	<i>min</i>	<i>min</i> ^[6]	<i>min</i> ^[2]	<i>min</i> ^[1]	<i>min</i> ^[2]	<i>min</i> ^[2]

Table 4. Security mechanism value assignment

Mechanism group	Security mechanism	M_{V_1}	M_{V_2}	M_{V_3}	M_{V_4}	M_{V_5}	Cost	Perf.
Encryption	SSH	0	0.6	0	0	0.6	30	30
	SSL	0	0.3	0	0	0.3	20	20
	TLS	0	0.8	0	0	0.8	40	20
Access control	Firewall	0.3	0.6	0.4	0	0	50	60
	AntiVirus	0.5	0.3	0.3	0	0	40	70
	Firewall & Antivirus	0.7	0.8	0.7	0	0	90	80
Hashing	MD5	0.3	0	0	0	0.3	10	20
	SHA2	0.6	0	0	0	0.6	30	20
	BLAKE2	0.8	0	0	0	0.8	40	20
Ident. management EMACS	Password	0.3	0	0	0	0	50	50
	Multi-factor	0.7	0	0	0	0	60	80
Ident. management SP IS	Password	0	0	0	0.3	0	50	50
	Smart Card	0	0	0.6	0	0	60	30

Scenario 1: The first scenario represents a simple optimisation process where all the scenario variables are set to be minimised, without any assigned priorities between them, as indicated the values of the second column in Table 3.

Table 5. Resulting system configurations per scenario

Mechanism	Scenario					
	1	2	3	4	5	6
Encryption	SSL	TLS	SSL	TLS	TLS	TLS
Access control	Firewall	Firewall & AntiVirus	AntiVirus	Firewall	Firewall & AntiVirus	Firewall
Hashing	MD5	BLAKE2	MD5	BLAKE2	BLAKE2	MD5
Ident. Mgmt EMACS	Password	Multi-factor Authent.	Password	Password	Multi-factor Authent.	Password
Ident. Mgmt SP IS	Password	SmartCard	Password	SmartCard	Password	SmartCard

The parameters of this scenario represent a case where the system's stakeholders require a system configuration which minimises the cost and added overhead while at the same time minimising the residual risk of all the potential threats. Such parameters lead to an implementation including, as shown in the second column of Table 5, SSL as the selected encryption technology, a firewall as an access control mechanism, MD5 as a hashing algorithm and Password Authentication Protocol as the authentication mechanism of choice for both the EMACS and the Swimming Pool Information System.

Scenario 2: The second scenario presents a variation of the first scenario where explicit priorities are set for the optimisation variables, as indicated by their superscript values, shown in the Scenario 2 column of Table 3. All variables are still set to be minimised but in this case the optimisation process prioritises the minimisation of the residual risks of the four identified threats (priorities [1] to [4] in the third column of Table 3) before optimising for the minimisation of the non-functional goals (priorities [5] for cost and [6] for performance). The solution identified in this case, as shown in the in the second column of Table 5, includes, TLS for encryption, both Firewall and Antivirus as access control mechanisms, BLAKE2 for hashing, multi-factor authentication for the EMACS system and Smart Card authentication for the Swimming Pool Information System.

Scenario 3: The third scenario is a variation of the previous scenario, where the top priority of the optimisation process is the minimisation of cost of the implementation. All variables are, once again, set to be minimised but the cost variable has the top priority, followed by the residual risks of all four threats and the added performance overhead, as indicated by the priority values of the fourth column of Table 3. The implementation produced as a result of such parameters includes security mechanism with the lowest cost value (i.e., SSL, Antivirus, MD5 and Password Authentication Protocol for both the EMACS and the Swimming Pool Information System).

Scenario 4: The fourth scenario is similar to the previous but, in this case, the added performance overhead has the top optimisation priority, with the residual risks following and the cost being the bottom priority. The identified solution,

as shown in the fifth column of Table 4, includes security mechanisms adding the least to the performance overhead of the system (i.e., TLS, Firewall, BLAKE2, Password Authentication Protocol and Smart Cards).

Scenario 5: In the fifth scenario, explicit thresholds have been set for the residual risks of threats 1 and 2, while also added cost and performance overhead have been assigned the top two priorities for the optimisation process. Therefore, this scenario represents a situation in which the system's stakeholders want to minimise both costs and added performance overheads, while at the same time requiring that the accepted residual risk of certain threats is limited to at most 25% of their initial inherent risk. As a result of the optimisation process, the implementation proposed contains TLS, both Firewall and Antivirus, BLAKE2, multi-factor authentication for the EMACS system and Password Authentication Protocol for the Swimming Pool Information System.

Scenario 6: The final scenario is similar to the previous one but, in this case, upper thresholds have been set for the accepted residual risks of all identified threats and the top optimisation priorities have been assigned to the non-functional aspects of the system (i.e., cost and performance). The maximum accepted values of residual risk for each threat, as indicated in the last column of Table 3, is expressed as a percentage of the initial inherent risk of each threat. The produced solution proposes the combination of TLS, firewall, MD5, Password Authentication Protocol and Smart Cards as the mechanisms of choice.

4.3 Discussion

The capability of our proposal to successfully adapt the system-at-hand in a diverse range of scenarios, indicates that (i) the proposed approach is adequately equipped to capture the contextual information necessary for quantitative risk assessment; and (ii) use it as the main input for an analysis process able to produce appropriate system configurations. The ability of the approach to accommodate any number of variables in the analysis process, both risk and soft-goal related, adds to its flexibility. For instance, in our case study, we identified four potential threats and two qualitative soft-goals for the whole system. Nevertheless, the application of the approach could easily scale in case of more threats, soft-goals and security mechanisms were identified. The same applies for the number of different security configurations identified through the presented scenarios. In our case study, we chose six scenarios in each of which the priorities or the maximum accepted values of the involved variables were different. We decided to do so in order to demonstrate the ability of the approach to generate different security configurations under diverse conditions. Nonetheless, any number of scenarios can be generated during the application of this approach, in order to reflect the needs and limitation of the system that is analysed.

Finally, it is worth indicating that certain aspects of the analysis in this case study were performed as a proof-of-concept and are not meant to be exhaustive. Therefore, a more complete security analysis could be supported by this approach if a more detailed constraint, threat or security mechanism elicitation process

takes place with the participation of security experts. The same is true for the value assignment of the variables related to the impact and likelihood of the identified vulnerabilities and the mitigation, cost and performance overhead of the identified security mechanisms. Nevertheless, the accuracy and completeness of the security analysis presented in the example used for this case study does not adversely affect the capabilities of the proposed approach.

5 Related Work

The work of Caillau *et al.* [5] introduces a probabilistic framework for goal-oriented risk analysis which performs quantitative reasoning using formal semantics in order to identify the effect of risks on the achievement of system goals. In [6], decision task models (DTMs), an extension of goal model diagrams, are introduced, which are able to capture temporal dimensions on goal tree structures, upon the nodes of which cost and benefit values can be attached. Based on such values and other formally defined constraints, an optimisation process can identify benefit-maximising system compositions. Our approach also makes use of constrained goal models for the performance of trade-off analysis but, in contrast with the above works, it has a clear information security orientation, as it is equipped with concepts and attributes which allows it to measure different aspects of risk and the effects of countermeasures on them.

Elahi *et al.* [10, 11] introduce a security-oriented approach for risk-aware trade off analysis, based on an extension of the i^* goal modelling framework. The notation introduced and the tool-supported analysis provided, however, are qualitative and therefore less fine-grained than the one proposed by our approach. A more implementation-oriented approach is presented by Yuan *et al.* [30], where architectural patterns are used for performing adaptations to the system according to the results of the evaluation of its security properties during runtime. It does not, however, elaborate on trade-offs between security and other system requirements as it is not meant to be utilised as a design-time approach since it requires a complete system architecture for its application.

The work of Pasquale *et al.* [24] introduces a requirements-driven approach for automated and quantitative security trade-off analysis through a sophisticated optimisation algorithm. Similarly, Aydemir *et al.* [2] propose a multi-objective, goal-oriented, risk modelling and analysis framework, which is based on constrained goal models and uses OptiMathSAT to identify optimal security countermeasures. As opposed to our approach, the focus of these works is exclusively on the design time system analysis, while their capabilities of capturing social aspects of the system are limited. To overcome these limitations, our approaches uses Secure Tropos as the basis of our analysis after extending it with concepts and attributes that allow capturing risk-related aspects. An attempt towards that end is presented by Matulevičius *et al.* [17] where Secure Tropos is conceptually aligned with the information system security risk management (ISSRM) reference model. As a result, its risk related conceptual limitations are identified, the most important of which being the lack of support for expressing

and quantitatively evaluating the concept of risk. The overcoming of such limitations is, therefore, amongst the motivating factors for the development of the approach presented in this work.

6 Conclusions and Future Work

In this paper we introduced a novel approach based on an extension of Secure Tropos with risk and CGM related concepts, capable of supporting quantitative risk assessment and trade-off analysis between security and other requirements. This is done by defining linear cost-functions to estimate the risk of each threat manifestation in our system and optimising various qualitative attributes, such as cost and performance. More specifically, we propose a framework which uses AHP to estimate the likelihood of threats to manifest and their impact of their manifestation at the system at hand. Our work also allows prioritising all the defined cost-functions provided by stakeholders and, with the use of an SMT/OMT reasoner, optimising them lexicographically by selecting a set of security mechanisms to be used as security countermeasures.

Future directions of this work include the exploration of other reasoners (e.g., Z3 [9]) which, in contrast with the OptiMathSAT, also support more complex, non-linear cost-functions. Another area in need of further exploration is the framework's current reliance on humans in the loop of the adaptation process, in order to assess the levels of risks and propose security countermeasures. A possible direction on further automating the risk management process, is investigating the processes and the information required for assessing the impact of a threat and the effectiveness of a security mechanism.

References

1. Argyropoulos, N., Márquez Alcañiz, L., Mouratidis, H., Fish, A., Rosado, D.G., de Guzmán, I.G.-R., Fernández-Medina, E.: Eliciting security requirements for business processes of legacy systems. In: Ralyté, J., España, S., Pastor, Ó. (eds.) PoEM 2015. LNBIP, vol. 235, pp. 91–107. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25897-3_7
2. Aydemir, F.B., Giorgini, P., Mylopoulos, J.: Multi-objective risk analysis with goal models. In: 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), pp. 1–10. IEEE (2016)
3. Blakley, B., McDermott, E., Geer, D.: Information security is information risk management. In: Proceedings of the 2001 Workshop on New Security Paradigms, pp. 97–104. ACM (2001)
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: an agent-oriented software development methodology. *Auton. Agent. Multi-Agent Syst.* **8**(3), 203–236 (2004)
5. Cailliau, A., Van Lamsweerde, A.: A probabilistic framework for goal-oriented risk analysis. In: 2012 20th IEEE International Requirements Engineering Conference (RE), pp. 201–210. IEEE (2012)

6. Chatzikonstantinou, G., Athanasopoulos, M., Kontogiannis, K.: Task specification and reasoning in dynamically altered contexts. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 625–639. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_42
7. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-functional Requirements in Software Engineering. Springer, Boston (2000). <https://doi.org/10.1007/978-1-4615-5269-7>
8. Dardenne, A., Van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. *Sci. Comput. Program.* **20**(1–2), 3–50 (1993)
9. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
10. Elahi, G., Eric, S.: A semi-automated tool for requirements trade-off analysis. In: CAiSE Forum, pp. 9–16 (2011)
11. Elahi, G., Yu, E.: A goal oriented approach for modeling and analyzing security trade-offs. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 375–390. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75563-0_26
12. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Formal reasoning techniques for goal models. *J. Data Semant.* **1**(1), 1–20 (2003)
13. Islam, S., Fenz, S., Weippl, E., Mouratidis, H.: A risk management framework for cloud migration decision support. *J. Risk Financ. Manage.* **10**(2), 10 (2017)
14. ISO/IEC: 27005:2008 - Information technology - Security techniques - Information security risk management. Technical report, ISO/IEC (2008)
15. ISO/IEC: 27000:2014 - Information technology - Security techniques - Information security management systems - Overview and vocabulary. Technical report, ISO/IEC (2014)
16. Karlsson, J., Ryan, K.: A cost-value approach for prioritizing requirements. *IEEE Softw.* **14**(5), 67–74 (1997)
17. Matulevičius, R., Mayer, N., Mouratidis, H., Dubois, E., Heymans, P., Genon, N.: Adapting secure tropos for security risk management in the early phases of information systems development. In: Bellahsene, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 541–555. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69534-9_40
18. Mell, P., Scarfone, K., Romanosky, S.: A complete guide to the common vulnerability scoring system version 2.0. In: FIRST-Forum of Incident Response and Security Teams, pp. 1–23 (2007)
19. MITRE: Common attack pattern enumeration and classification, (CAPEC). <https://capec.mitre.org/>
20. Mouratidis, H., Argyropoulos, N., Shei, S.: Security requirements engineering for cloud computing: The secure tropos approach. In: Domain-Specific Conceptual Modeling, Concepts, Methods and Tools, pp. 357–380. Springer (2016)
21. Mouratidis, H., Giorgini, P.: Secure tropos: a security-oriented extension of the tropos methodology. *Int. J. Softw. Eng. Knowl. Eng.* **17**(2), 285–309 (2007)
22. Nguyen, C.M., Sebastiani, R., Giorgini, P., Mylopoulos, J.: Multi-objective reasoning with constrained goal models. *Requirements Eng.* (2016). <https://doi.org/10.1007/s00766-016-0263-5>
23. Open Web Application Security Project: Application threat modeling. Technical report, OWASP (2015)

24. Pasquale, L., Spoletini, P., Salehie, M., Cavallaro, L., Nuseibeh, B.: Automating trade-off analysis of security requirements. *Requirements Eng.* **21**, 481–504 (2015)
25. Saaty, T.L.: What is the analytic hierarchy process? In: Mitra, G., Greenberg, H.J., Lootsma, F.A., Rijkaert, M.J., Zimmermann, H.J. (eds.) *Mathematical Models for Decision Support*, pp. 109–121. Springer, Heidelberg (1988). https://doi.org/10.1007/978-3-642-83555-1_5
26. Sebastiani, R., Trentin, P.: OptiMathSAT: a tool for optimization modulo theories. In: Kroening, D., Păsăreanu, C.S. (eds.) *CAV 2015*. LNCS, vol. 9206, pp. 447–454. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_27
27. Stoneburner, G., Goguen, A., Feringa, A.: Risk management guide for information technology systems (NIST special publication 800–30). Tech. rep. (2002)
28. Vaidya, O.S., Kumar, S.: Analytic hierarchy process: an overview of applications. *Eur. J. Oper. Res.* **169**(1), 1–29 (2006)
29. Viduto, V., Maple, C., Huang, W., Bochenkov, A.: A multi-objective genetic algorithm for minimising network security risk and cost. In: 2012 International Conference on High Performance Computing and Simulation (HPCS), pp. 462–467. IEEE (2012)
30. Yuan, E., Malek, S., Schmerl, B., Garlan, D., Gennari, J.: Architecture-based self-protecting software systems. In: *Proceedings of the 9th International ACM SIGSOFT Conference on Quality of Software Architectures*, pp. 33–42. ACM (2013)



Erratum to: Enforcing Memory Safety in Cyber-Physical Systems

Eyasu Getahun Chekole^{1,2(✉)}, John Henry Castellanos¹,
Martín Ochoa¹, and David K. Y. Yau^{1,2}

¹ Singapore University of Technology and Design, Singapore, Singapore
eyasu_chekole@mymail.sutd.edu.sg

² Advanced Digital Sciences Center, Singapore, Singapore

Erratum to:

Chapter “Enforcing Memory Safety in Cyber-Physical
Systems” in: S. K. Katsikas et al. (Eds.):

Computer Security, LNCS 10683,

https://doi.org/10.1007/978-3-319-72817-9_9

The original version of this chapter unfortunately contained a mistake. The first author’s name was incorrect. The correct name should read: Eyasu Getahun Chekole.

The updated online version of this chapter can be found at
https://doi.org/10.1007/978-3-319-72817-9_9

© Springer International Publishing AG 2018
S. K. Katsikas et al. (Eds.): CyberICPS 2017/SECPRE 2017, LNCS 10683, p. E1, 2018.
https://doi.org/10.1007/978-3-319-72817-9_18

Author Index

- Agrawal, Anand 110
Ahmed, Chuadhry Mujeeb 177
Alshammari, Majed 189
Amichay, Ori 93
Angelopoulos, Konstantinos 262
Argyropoulos, Nikolaos 262
Armknecht, Frederik 3
- Bar, Ofer 93
- Castellanos, John Henry 127
Chekole, Eyasu Getahun 127
Chitrakar, Ambika Shrestha 163
- Dagnat, Fabien 63
Dashti, Mohammad Torabi 250
Diamantopoulou, Vasiliki 210
Dyrkolbotn, Geir Olav 147
- Fish, Andrew 262
Foley, Simon N. 229
Fontaine, Caroline 63
- Gomez, Laurent 3
Govil, Naman 110
- Helkala, Kirsi 147
- Jelacic, Bojan 77
Jøsok, Øyvind 147
- Kleinmann, Amit 93
Knox, Benjamin J. 147
- La Marra, Antonio 35
Lendak, Imre 77
Lev, Leonid 93
Lugo, Ricardo G. 147
- Márquez, José 3
Martinelli, Fabio 35
- Mikhalev, Vasily 3
Mori, Paolo 35
Mouratidis, Haralambos 210, 262
Murguia, Carlos 177
- Naessens, Vincent 19
- Ochoa, Martín 127
- Pavlidis, Michalis 210
Petrović, Slobodan 163
- Qadeer, Rizwan 177
- Radomirović, Saša 250
Raes, Vincent 19
Rizos, Athanasios 35
Rooney, Vivien M. 229
Rosic, Daniela 77
Ruths, Justin 177
- Sandberg, Christian 47
Saracino, Andrea 35
Scandariato, Riccardo 47
Simpson, Andrew 189
Stanojevic, Marina 77
Stoja, Sebastijan 77
Sultan, Bastien 63
Sütterlin, Stefan 147
Svendsen, Nils Kalstad 147
- Tenenbaum, David 93
Tippenhauer, Nils Ole 110
Tuma, Katja 47
- Vossaert, Jan 19
- Widman, Mathias 47
Wool, Avishai 93
- Yau, David K. Y. 127