Sokratis Katsikas
Costas Lambrinoudakis
Steven Furnell (Eds.)

Trust, Privacy and Security in Digital Business

LNCS 9830

13th International Conference, TrustBus 2016
Porto, Portugal, September 7–8, 2016
Proceedings

Springer

Lecture Notes in Computer Science 9830

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

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

Sokratis Katsikas · Costas Lambrinoudakis
Steven Furnell (Eds.)

Trust, Privacy and Security in Digital Business

13th International Conference, TrustBus 2016
Porto, Portugal, September 7–8, 2016
Proceedings

Springer

Editors
Sokratis Katsikas
Norwegian University of Science
and Technology
Gjøvik
Norway

Costas Lambrinoudakis
University of Piraeus
Piraeus
Greece

Steven Furnell
Plymouth University
Plymouth
UK

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-44340-9 ISBN 978-3-319-44341-6 (eBook)
DOI 10.1007/978-3-319-44341-6

Library of Congress Control Number: 2015946097

LNCS Sublibrary: SL4 – Security and Cryptology

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

Preface

This book presents the proceedings of the 13th International Conference on Trust, Privacy and Security in Digital Business (TrustBus 2016), held in Porto, Portugal, during September 7–8, 2016. The conference continues from previous events held in Zaragoza (2004), Copenhagen (2005), Krakow (2006), Regensburg (2007), Turin (2008), Linz (2009), Bilbao (2010), Toulouse (2011), Vienna (2012), Prague (2013), Munich (2014), and Valencia (2015).

The advances in the Information and Communication Technologies (ICT) have raised new opportunities for the implementation of novel applications and the provision of high-quality services over global networks. The aim is to utilize this "information society era" for improving the quality of life for all citizens, disseminating knowledge, strengthening social cohesion, generating earnings, and finally ensuring that organizations and public bodies remain competitive in the global electronic marketplace. Unfortunately, such a rapid technological evolution cannot be problem-free. Concerns are raised regarding the "lack of trust" in electronic procedures and the extent to which "information security" and "user privacy" can be ensured.

TrustBus 2016 brought together academic researchers and industry developers who discussed the state of the art in technology for establishing trust, privacy, and security in digital business. We thank the attendees for coming to Porto to participate and debate the new emerging advances in this area.

The conference program included a keynote and four technical papers sessions that covered a broad range of topics, from security, privacy, and trust in eServices, to security and privacy in cloud systems and mobile environments. The conference attracted many high-quality submissions, each of which was assigned to four referees for review and the final acceptance rate was 43 %.

We would like to express our thanks to the various people who assisted us in organizing the event and formulating the program. We are very grateful to the Program Committee members and the external reviewers, for their timely and rigorous reviews of the papers. Thanks are also due to the DEXA Organizing Committee for supporting our event, and in particular to Gabriela Wagner for her help with the administrative aspects.

Finally, we would like to thank all of the authors who submitted papers for the event and contributed to an interesting technical program.

September 2016

Sokratis Katsikas
Costas Lambrinoudakis
Steven Furnell

Organization

General Chair

Steven Furnell	Plymouth University, UK

Program Committee Co-chairs

Sokratis Katsikas	Norwegian University of Science and Technology - NTNU, Norway
Costas Lambrinoudakis	University of Piraeus, Greece

Program Committee

Aggelinos, George	University of Piraeus, Greece
Agudo Ruiz, Isaac	University of Malaga, Spain
Rudolph, Carsten	Monash University, Australia
Casassa Mont, Marco	HP Labs Bristol, UK
Chadwick, David	University of Kent, UK
Chu, Cheng-Kang	Huawei International, Singapore
Clarke, Nathan	University of Plymouth, UK
Cuppens, Frederic	ENST Bretagne, France
De Capitani di Vimercati, Sabrina	Università degli Studi di Milano, Italy
Domingo-Ferrer, Josep	Rovira i Virgili University, Spain
Drogkaris, Prokopis	University of Piraeus, Greece
Eloff, Jan	University of Pretoria, South Africa
Fernandez, Eduardo B.	Florida Atlantic University, USA
Fernandez-Gago, Carmen	University of Malaga, Spain
Ferrer Gomila, Jose Luis	University of Balearic Islands, Spain
Fischer-Huebner, Simone	Karlstad University, Sweden
Foresti, Sara	Università degli Studi di Milano, Italy
Fuß, Jürgen	University of Applied Sciences Upper Austria at Hagenberg, Austria
Geneiatakis, Dimitris	Aristotle University of Thessaloniki, Greece
Gritzalis, Dimitris	Athens University of Economics and Business, Greece
Gritzalis, Stefanos	University of the Aegean, Greece
Hansen, Marit	Independent Center for Privacy Protection Schleswig-Holstein, Germany
Kalloniatis, Christos	University of the Aegean, Greece
Karyda, Maria	University of the Aegean, Greece
Kesdogan, Dogan	University of Regensburg, Germany

Kokolakis, Spyros	University of the Aegean, Greece
Kowalski, Stewart	Norwegian University of Science and Technology, Norway
Lioy, Antonio	Politecnico di Torino, Italy
Lopez, Javier	University of Malaga, Spain
Markowitch, Olivier	Université Libre de Bruxelles, Belgium
Marsh, Stephen	University of Ontario, Institute of Technology, Canada
Martinelli, Fabio	CNR, Italy
Matyas, Vashek	Masaryk University, Czech Republic
Megias, David	Open University of Catalonia, Spain
Mitchell, Chris	Royal Holloway, University of London, UK
Mouratidis, Haralambos	University of Brighton, UK
Olivier, Martin S.	University of Pretoria, South Africa
Oppliger, Rolf	eSecurity Technologies, Switzerland
Papadaki, Maria	Plymouth University, UK
Pashalidis, Andreas	BSI, Germany
Patel, Ahmed	Universiti Kebangsaan Malaysia, Malaysia
Pernul, Guenther	University of Regensburg, Germany
Posegga, Joachim	University of Passau, Germany
Quirchmayr, Gerald	University of Vienna, Austria
Rizomiliotis, Panagiotis	University of the Aegean, Greece
Roman Castro, Rodrigo	University of Malaga, Spain
Ruland, Christoph	University of Siegen, Germany
Samarati, Pierangela	Università degli Studi di Milano, Italy
Skarmeta, Antonio F.	University of Murcia, Spain
Teufel, Stephanie	University of Fribourg, Switzerland
Theoharidou, Marianthi	European Commission - Joint Research Centre, Italy
Tjoa, A Min	Technical University of Vienna, Austria
Tomlinson, Allan	Royal Holloway, University of London, UK
Tsochou, Aggeliki	Ionian University, Greece
Weippl, Edgar	SBA Research and Vienna University of Technology, Austria
Xenakis, Christos	University of Piraeus, Greece

Contents

Security, Privacy and Trust in eServices

A Framework for Systematic Analysis and Modeling of Trustworthiness Requirements Using i* and BPMN

Nazila Gol Mohammadi(✉) and Maritta Heisel

Paluno - The Ruhr Institute for Software Technology,
University of Duisburg-Essen, Essen, Germany
{nazila.golmohammadi,maritta.heisel}@paluno.uni-due.de

Abstract. New technologies like cloud computing and new business models bring new capabilities for hosting and offering complex collaborative business operations. However, these advances can also bring undesirable side-effects, e.g., introducing new vulnerabilities and threats caused by collaboration and data exchange over the Internet. Hence, users become more concerned about the trust, e.g., trust in services for critical business processes with sensitive data. Since trust is subjective, trustworthiness requirements for addressing trust concerns are difficult to elicit, especially if there are different parties involved in the business process. In this paper, we propose a user-centered trustworthiness requirement analysis and modeling framework. Using goal models for capturing the users' trust concerns can motivate design decisions with respect to trustworthiness. We purpose integrating the subjective trust concerns into goal models and embedding them into business process models as objective trustworthiness requirements. This paper addresses the gap in considering trustworthiness requirements during automation (in providing supporting software) of business processes. We demonstrate our approach on an application example from the health-care domain.

Keywords: Trust · Trustworthiness requirements · Business process modeling · Requirements engineering · Goal modeling

1 Introduction

Advances in new technologies such as cloud, social and mobile computing have been an important enabler for developing business information systems that support nowadays' complex businesses. These new technologies bring new capabilities for hosting and offering highly dynamic and collaborative business processes, e.g., health-care services via Internet in the medical domain. The trustworthiness of business information systems that support collaborative business processes is a key factor for promoting such collaboration and consequently the adoption of these systems. Trustworthiness requirements must be assured, in order to meet users' trust concerns. To support users' confidence (leading to business services adoption), the right mechanisms should be put into place. Trustworthiness

S. Katsikas et al. (Eds.): TrustBus 2016, LNCS 9830, pp. 3–18, 2016.
DOI: 10.1007/978-3-319-44341-6_1

requirements should be in accordance with end-users' trust concerns. Furthermore, business processes and their involved software systems and services need to be made trustworthy to mitigate the risks of engaging those systems.

For being trustworthy, business information systems should fulfill a variety of qualities and properties that depend on the application and its domain [9]. For instance, organizations as users require confidence about their business-critical data, whereas an elderly person using a health-care service may be more concerned about reliability and usability. The traditional development methodologies do not respect users' trust concerns in dynamic, heterogeneous, and distributed settings. Recently, innovative technologies like trustworthiness-by-design methodologies [6], are attracting researchers' attention. Requirements engineering is a critical activity in such "by-design" methodologies. However, there only exists a small set of well-accepted requirement refinement methods and complementary decision support (supporting design decisions), which can be applied in a systematic way for considering trustworthiness [3]. We believe that trustworthiness of business systems is strongly dependent on their development processes, especially the elaboration of trustworthiness requirements during the requirement engineering phase [6].

To bridge the gap between requirements and design artifacts in addressing trust concerns, we propose a framework to specify and analyze trustworthiness requirements in a systematic and iterative way. Trust concerns are identified and addressed in the goal models by trustworthiness goals. Consequently, trustworthiness requirements are refined in goal models iteratively in combination with the business process models defined for satisfying the goals. In this way, it is ensured that trustworthiness requirements will not be violated or ignored, while developing or implementing the activities, resources and data-objects involved in the business processes. We propose a conceptual model and a framework for systematic analysis, documentation and modeling trustworthiness requirements in a user-centered manner. The paper aims at bringing together trustworthiness requirements analysis with regard to trust concerns and thereafter building trustworthiness properties into underlying systems for performing business processes. Our objectives are to analyze and specify trustworthiness requirements in the business process models to support the process designer and tool developers in fulfilling trustworthiness requirements and a later evaluation of them. We use i* [23] for goal-modeling and **B**usiness **P**rocess **M**odel and **N**otation (BPMN) [13] for modeling business processes. The main challenges that we discovered based on an analysis of the state of the art are a lack of concepts relevant for trustworthiness (e.g., delegations) and a lack of inter-model consistency checks between BPMN and i* models. Goal models combined with business process models specify how business processes fulfill the trustworthiness goals. Our framework makes it possible to document the trustworthiness requirements together with the corresponding knowledge of the system's context. Furthermore, it supports the process of refining (soft-) goals right up to the elicitation of corresponding trustworthiness requirements.

Our approach is beneficial for the decision support during run-time adaptation as well. In an uncertain and changing environment, business processes

are continuously optimized, e.g., via service substitution. To respect the overall trustworthiness level, quality trade-offs should respect trustworthiness requirements. The business process models enhanced with trustworthiness properties are useful information during the run-time as well.

The remainder of this paper is structured as follows: In Sect. 2, we explain the fundamentals of our framework. Section 3 presents our framework for combining goal models and business process modeling to support eliciting and analyzing trustworthiness requirements and embedding them in business process models. We demonstrate the application of our framework on a case study inspired from the EU project OPTET[1] in Sect. 4. Section 5 discusses related work. Finally, Sect. 6 gives a conclusion and sketches future work.

2 Background and Fundamentals

In this section we briefly introduce the fundamental techniques and concepts for the framework that is described in Sect. 3.

Trust and Trustworthiness. *Trust* is defined as a "bet" about the future contingent actions of a system [19]. The components of this definition are belief and commitment. There is a belief that placing trust in a software or a system will lead to a good outcome. Then, the user commits the placing of trust by taking an action by using a business process. This means when some users decide to use a service, e.g., a health-care service on the web, they are confident that it will meet their expectations. Trust is subjective and different from user to user. For instance, organizations require confidence about their business-critical data, whereas an elderly person using a health-care service (end-users) may be more concerned about usability. These concerns manifest as *trustworthiness requirements*.

Trustworthiness properties are qualities of the system that potentially influence trust in a positive way. The term trustworthiness is not used consistently in the literature. Trustworthiness has sometimes been used as a synonym for security and sometimes for dependability. However, security is not the only aspect of trustworthiness. Some approaches merely focus on single trustworthiness characteristics, e.g., security or privacy. However, trustworthiness is rather a broad-spectrum term with notions including reliability, security, performance, and usability as parts of trustworthiness properties [11]. Trustworthiness is domain and application dependent. For instance, in health-care applications, the set of properties which have primarily been considered consists of availability, confidentiality, integrity, maintainability, reliability and safety, but also performance and timeliness.

Business Process Modeling Using BPMN. A business process is a specific ordering of activities across time and place, with a start, an end, and

[1] http://www.optet.eu/.

clearly defined inputs and outputs. A business process model is the representation of the activities, documents, people and all the elements involved in a business process, as well as the execution constraints between them [18]. By using business process modeling, different information can be captured such as organizational, functional, informational, behavioral and context information. The organizational information focuses on the actors and their activities. The functional information describes the process element activity which is being performed during a business process execution. A resource can either be a human resource or a technical resource, such as tools or a service used in performing an activity, or informational resources, such as data. The business process models also represent how the informational resources are manipulated in a process. The behavioral information includes the time aspects of activities by focusing on when activities are performed and when they are sequenced. We can show control flow and data flow in business process models.

BPMN [13] is a standard for modeling business processes, which is broadly extended and used widely in both, industry and research. The most important BPMN elements are shown in Fig. 6.

Goal Modeling. In requirements engineering, goal modeling approaches have gained considerable attention in varying contexts. These approaches aim at capturing the rationale of the software system development. A goal model defines organization goals and the tasks necessary to achieve these goals. Thus, goal models relate the high-level goals of an organization to low-level system requirements. Goals can be classified into two different categories: hard-goals and soft-goals. Hard-goals may refer to the functional properties of the system behavior, whereas soft-goals represent quality preferences of the stakeholders. There exist a number of different goal modeling languages used in requirements engineering. We use i* in our analysis due to its comprehensiveness.

The i* notation was developed with the purpose of modeling and reasoning within an organizational environment and its information systems [23]. It consists of two main models, a **S**trategic **D**ependency **M**odel (SDM) and a **S**trategic **R**ationale **M**odel (SRM). The SDM (cf. Fig. 5) is used to express strategic relationships among different actors in an organizational context. The SRM (cf. Fig. 7) captures both an internal view of each actor and external relations among actors. The main concepts used in i* models are actors, goals, tasks, resources and soft goals. An actor is a role who carries out a task to achieve a certain goal. A resource is an object that is needed to complete a goal or perform some task. The following dependencies can be defined in i*: goal, soft-goal, task or resource dependencies (cf. Fig. 5). For the internal view of an actor in SRM, the links are as follows: means-ends, task decomposition and contribution (cf. Fig. 7).

3 Framework for Systematic Analysis and Modeling of Trustworthiness Requirements

Our proposed goal-business process model is employed to decompose high-level goals into low-level goals. We shape and structure our framework (shown in Fig. 1)

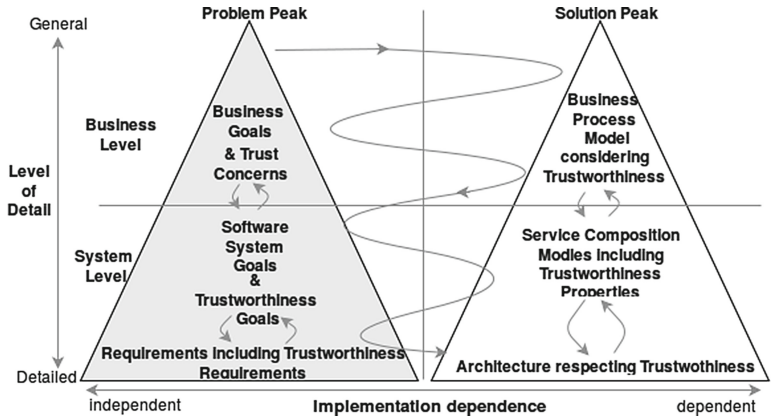


Fig. 1. Overview of proposed framework inspired by [12]

based on the twin peaks model [12]. The cornerstone of embedding the development of business information systems in the twin peak model is that requirements engineers and developers build a system's requirements and its architecture specification concurrently and iteratively. The same applies to our proposed approach for the analysis of trustworthiness requirements and integrating them into business models. The business processes are defined to fulfill goals with trustworthiness embedded into the business processes.

The major method of our framework for eliciting and refining trustworthiness requirements is the combination of business process modeling (to show how, solution peak) using BPMN and goal models (to say what, problem peak) in i*. The details about the conceptual model of the framework and method are presented in the following sections.

3.1 Conceptual Model

We use the basis described in Sect. 2 to analyze how the described goal modeling components align with process model components. We analyze the ability of goal modeling in assisting the business process models in enabling trustworthiness properties. We use certain concepts to facilitate the analysis of business process models respecting trustworthiness. The relationship between these concepts is depicted in a conceptual model shown in Fig. 2 as **U**nified **M**odeling **L**anguage (UML) class diagram. The conceptual model depicts the basic concepts of our approach.

A *trustworthiness goal* is a special *goal* that addresses the trust concerns of users. A trustworthiness goal is satisfied by *trustworthiness requirements*, which can be realized by more concrete *trustworthiness properties*. *Actors* have goals that can be satisfied in a *business process*. A business process consists of *business process elements* (a set of activities, events, and involved resources). Here, activities, resources, or events are more concrete *business process elements*. An actor

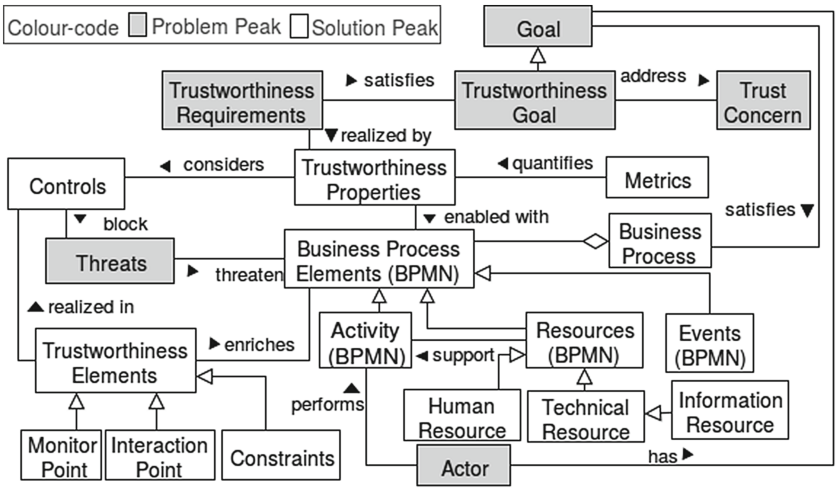


Fig. 2. Conceptual model of our proposed framework and the method

performs an *activity*. An activity is supported by resources. For instance, an activity consumes data objects (information resource) as input, or it produces output, or technical resources support performing an activity such as software services and applications. We use the term *business process element* to distinguish between generic types of BPMN and concrete *trustworthiness elements* (our extension to BPMN in [7]).

This paper focuses on the part of the framework for analyzing and addressing the end-users' trust concerns, using goal and business process models. We defined *trustworthiness elements* to enrich *business process elements* by defining *monitor point* or *interaction point* or *constraints* on *business process elements*. For instance, a trustworthiness element can be a trustworthiness-specific activity (e.g., notifications for satisfying transparency) or a monitoring point where we can specify which part of the process needs to be monitored during run-time and what the desired behaviors are. This will serve to derive trustworthiness requirements in the form of commitments reached among the participants for the achievement of their goals. The precise specification of our BPMN extension is described in another paper [7].

A *threat* is a situation or event that, if active at run-time, could undermine the trustworthiness by altering the behavior of involved resources or service in the process. *Controls* aim at blocking threats. *Metrics* are used as functions to quantify trustworthiness properties. A metric is a standard way for measuring and quantifying certain trustworthiness properties as more concrete quality properties of an element [4,9]. Trustworthiness elements realize the control in terms of defining elements which address the trustworthiness, e.g., an additional activity can be defined to block a threat to privacy. These additional activities could involve documenting or triggering a notification upon a delegating case of a patient to another authority, or an engagement of a new service from a new third party.

3.2 The Method for Systematic Analysis of Trustworthiness Requirements

Figure 3 gives an overview of the steps of the method and their inputs and outputs. The steps are as follows:

Step 1 - Context Analysis: The first step is concerned with identifying the participants and initial context information. This can also be captured in a context model. The context information provides an overview of the process, as well.

Step 2 - Set Up Goal Model: This step is concerned with setting up the goal model by capturing the major intentions of the involved participants/stakeholders. The goals are captured either by interviewing involved stakeholders or are based on expertise of a requirements engineer or business engineer at the business level. We start with high-level goals, and then refine them within the problem (requirement) peak. We model and document the goals using i* with SDM and SRM models.

Step 3 - Set Up Business Process Models: As input the SDM and SRM models are used. We select a specific goal from SDM. For satisfying the selected goal we set up a business process model. As notation, we use BPMN. To create the business process model we use information shown in the SDM and SRM. Using SDM, the dependency between roles and other goals can be anlysed. SRM models give insight into the resources and activities. The business process model for a specific goal selected from SDM models will visualize the control and data flow between identified tasks, used resources and involved actors.

Step 4 - Identify Trust Concerns: Trust concerns of end-users and their dependencies on other participants in the business are identified. Trust concerns can be collected either by interviewing involved end-users/consumers or are based on the expertise of a requirements engineer. Trust concerns are subjective. To support this step (especially considering subjectiveness of trust), a questionnaire is provided in our previous work [8].

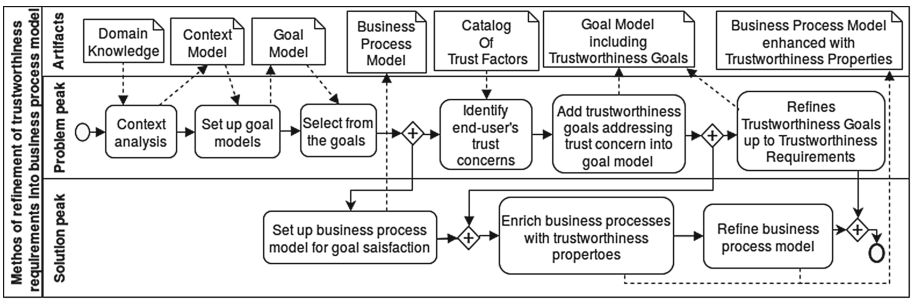


Fig. 3. The method for analysis of trustworthiness requirements and including trustworthiness properties into the business process models

Step 5 - Goal Model Including Trustworthiness Goals: Based on trust concerns, we refine the goal model with the trustworthiness goals and their relation to the other goals (conflicts or positive influences). The trustworthiness goals include the purpose of the building of trustworthiness properties into the system under development. To support this step, a catalogue of trustworthiness attributes which contribute to mitigate trust concerns is provided in our previous work [9].

Step 6 - Business Process Model Including Trustworthiness Properties: Enhance a business process model by adding trustworthiness properties which fulfill the trustworthiness goals. For supporting this step, we provide the new trustworthiness elements (cf. Fig. 2). The business process model from step 3 is analysed by identifying which business process elements are related to the identified trustworthiness goals from step 5. The relation of trustworthiness goals in the goal model to the other goals from step 5 assists this step.

Step 7 - Refinement of Goal Model (Problem Peak): Refine goals and trustworthiness goals further to obtain user-centered trustworthiness requirements on resources and tasks. This refinement is performed within the problem peak. However, based on the output of this step revisions of business process models can be necessary.

Step 8 - Refinement of Business Process Model (Solution Peak): Detail business processes by including trustworthiness properties on resources, activities, etc. for satisfying trustworthiness requirements. This refinement is performed within the solution peak. However, based on the output of this step revisions of goal models can be necessary.

4 Application Example

This section demonstrates our approach of eliciting and refining trustworthiness requirements and specifying trustworthiness properties on business process elements. The example stems partially from the experience that the first author gained during the OPTET project on an **A**mbient **A**ssisted **L**iving (AAL) system.

Motivating Scenario. In our scenario, Alice is an elderly person who lives alone in her apartment. She does not feel comfortable after a heart attack. She was unconscious in her home for several hours. Alice has been informed that there are some AAL services available in the marketplace. She considers using one of those services to avoid similar incidents in the future. She desires an AAL service that will suit her specific needs. We illustrate, in Fig. 4, a general approach using supporting tools and provided apps to perform the activities. We assume that some of these software services are to be built by software developers, who will also benefit from the results of our work in developing trustworthy apps, software services, etc.

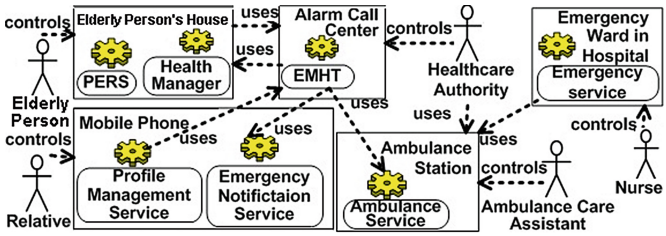


Fig. 4. Part of home monitoring system for handling health-care cases inspired by [5]

Step 1 - Context Analysis: We will focus on a ***H****ome* ***M****onitoring* ***S****ystem* (HMS) for incident detection and detection of abnormal situations to prevent emergency incidents. The *HMS* allows elderly people in their homes to call for help in case of emergency situations. Furthermore, HMS analyzes the elderly person's health status for preventing incidents in the first place. The incidents are reported to an *Alarm Call Center* that, in turn, reacts by, e.g., sending out ambulances or other medical caregivers, and notifying the elderly person's relatives. For preventing emergency situations, the vital signs of the elderly person are diagnosed in regular intervals to reduce hospital visits and falls. Figure 4 shows an exemplary design-time system model including physical, logical, and human resources/assets. Using this system, an elderly person uses a ***P****ersonal* ***E****mergency* ***R****esponse* ***S****ystem (PERS)* device to call for help, which is then reported to the alarm call center that uses an ***E****mergency* ***M****onitoring and* ***H****andling* ***T****ool (EMHT)* to visualize, organize, and manage emergency incidents. Furthermore, elderly persons are able to use a ***H****ealth* ***M****anager (HM)* app on their smart device for organizing their health status like requesting health-care services or having an overview regarding their medication or nutrition plan. The EMHT is a software service hosted by the alarm call center that, in turn, is operated by a health-care authority. *Emergency notification* and *Ambulance Service*, which run on mobile phones of relatives, or *Ambulance Stations* respectively, are called in order to require caregivers to provide help. An Ambulance Service is requested in case an ambulance should be sent to handle an emergency situation. The other case is that, based on analyzed information sent to the *EMHT*, an abnormal situation is detected and further diagnoses are necessary. Therefore, the elderly person will get an appointment and notifications for a tele-visit in her *HM* app.

Step 2 - Set Up Goal Model: Figure 5 captures the goals of different participants and their dependencies on each other or the realization of the goals. This is done based on expertise of a requirements engineer and the knowledge gained during the context analysis like interviews. Here, we only focus on the *Elderly person* and the *Alarm Call Center*. The *Ambulance Station* has also been considered, because for handling the emergency cases the alarm call center is dependent on the ambulance as a resource.

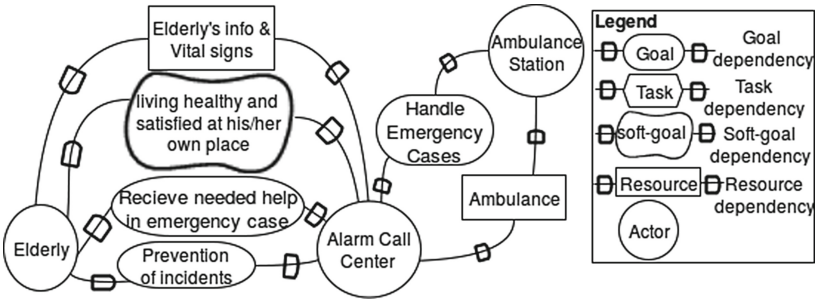


Fig. 5. Simplified SDM with the dependencies between identified participants

Additional to SDM presented in Fig. 5, we have further SRM models which gives more detail on tasks, resources and soft-goals within the actor boundaries. As an example, one can consider the SRM model in Fig. 7. In this step we have only the white-coloured elements of that SRM.

Step 3 - Set Up Business Process Model: Figure 6 illustrates and exemplifies the typical steps that, e.g., caregivers in an alarm center have to take once they analyzed that the health record of an elderly person deviates from the normal situation and further examination is needed. This business process model targets the satisfaction of *reducing hospital visits* and the *prevention of incidents* goals (cf. Fig. 5). The process starts by *analysing the elderly person's vital signs in the last 7 days*. These data is examined by a physician, who decides whether the elderly person is healthy or if additional examination needs to be undertaken. In the former case, the physician fills out the examination report. In the latter case, a *tele-visit* is performed by this physician in which the physician informs the elderly person about examination and necessary treatment. An *examination order* is placed by the physician. The physician sends out a request.

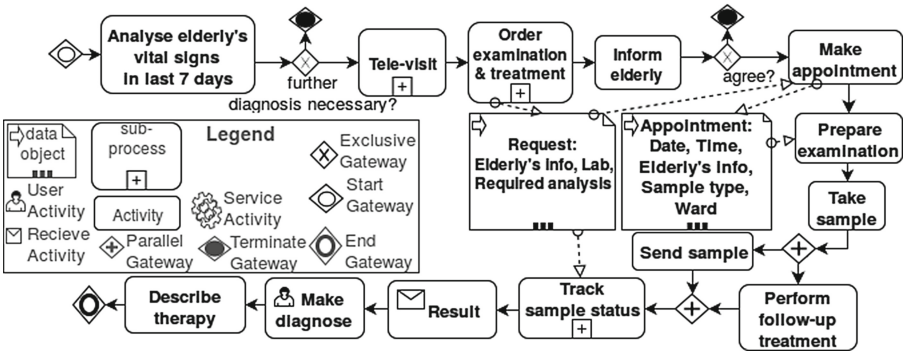


Fig. 6. Exemplary business process model for preventing emergency cases and reducing hospital visits

This request includes information about the elderly person, the required examination and possible labs. Furthermore, an appropriate appointment should be arranged. The process continues for taking a sample and validating this. Eventually, the physician from the *Alarm Call Center* should get the result in order to make the diagnosis and prescribing the medication.

Step 4 - Identify Trust Concerns: Alice is concerned about the fact whether she will really receive the emergency help if a similar situation happens again (heart attack experience). Alice is informed that by using the HMS she can have regular diagnoses which can prevent frequent hospital visits. However, Alice is concerned whether she will be able to use the service in a proper way. She is also concerned about who can get access to the data about her diseases or life habits. She indicates that she would only like her regular nurse and doctor to be able to see her history and health status.

Step 5 - Goal Model Including Trustworthiness Goals: Based on the trust concerns and the application domain and considering necessary legislation, a requirement engineer will add trustworthiness goals to the goal model. The existing goal-based refinement techniques will be applied to refine these trustworthiness goals into trustworthiness requirements. Considering the health-care domain, reliability, availability, usability, raising awareness and privacy (providing guidance and user's data protection) is a crucial issue related to trustworthiness [1]. For instance, electronic medical transactions require the transmission of personal and medical information over insecure channels, e.g., the Internet. Patients' profiles document the medical behavior of patients, or even include sensitive information, e.g., their medical history. Considering trustworthiness of a health-care application, one can consider a vector of multiple trustworthiness goals. They either address the fulfillment of the mission, e.g., reliability, availability when the patient needs help, correctness of prescribed therapy or address it from a privacy perspective. The gray-coloured soft-goals in Fig. 7 are the trustworthiness goals added to the goal model in this step. The initial SRM of the elderly person and the alarm call center contain only the white-coloured elements of Fig. 7.

Step 6 - Business Process Including Trustworthiness Properties: Figure 8 illustrates the enriched business process model with the trustworthiness requirements satisfying *reliability and privacy* (cf. Fig. 7). In particular, we exemplify the typical steps that a human resource (e.g., caregiver in alarm call center) has to take or properties that a non-human resource needs to have in order to contribute to trustworthiness. We start with the activity to *analyse the history of the vital signs* of the elderly person in the last seven days. This activity may detect a risk in her health status. For addressing the trust concerns of the elderly person related to her confidence that she is not left alone and will get the needed health care in case when necessary, as well as privacy-related concerns, the following trustworthiness requirements are specified: The elderly person should receive a regular notification that informs her about the diagnoses that are performed on her vital signs. In Fig. 8 it is added as a trustworthiness-related activity, namely *"Notify elderly"*.

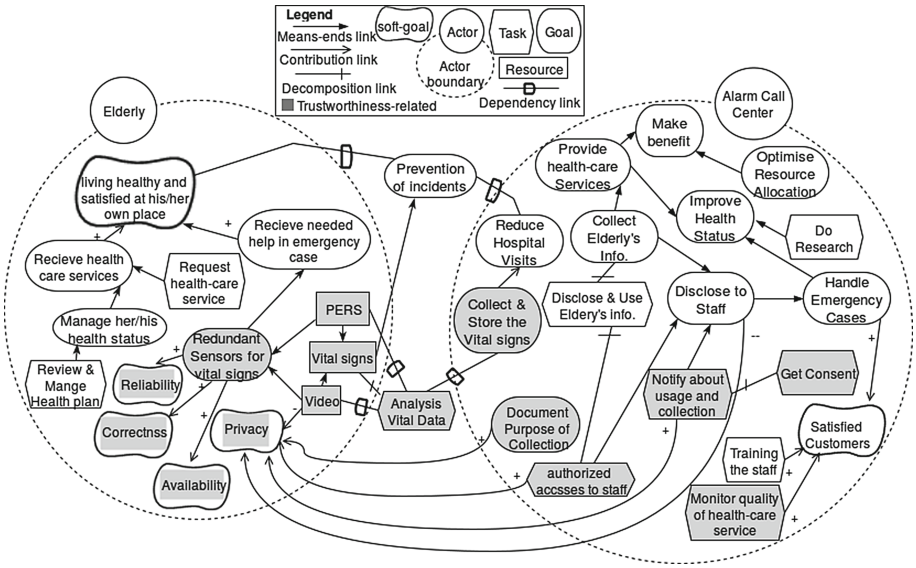


Fig. 7. Simplified SRM including trustworthiness goals considering trust concerns

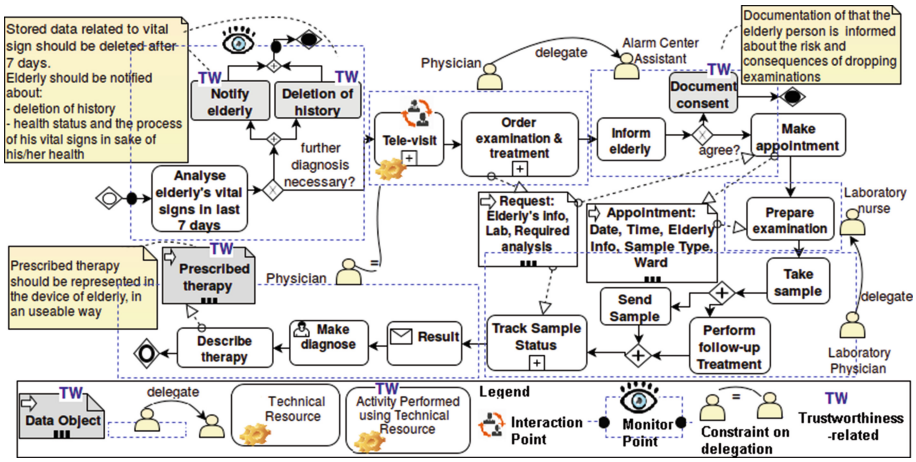


Fig. 8. Exemplary business process model enriched with trustworthiness requirements

This activity contributes to make her confident that she is not left alone without care. Because of privacy, in case no further diagnosis is necessary, the history should be deleted. The *Deletion of history* activity is also a trustworthiness-related activity added to the initial business process. This part of the business process is annotated as relevant for monitoring at run-time.

If a risk to the elderly person's health status is detected, a *tele-visit* is offered. This activity is an interaction point supported by the HM app as technical

resource (cf. Fig. 8, tele-visit activity performed by a physician). The trustworthiness properties for this interaction point are usability, response time, etc. In case of necessity for further examination the elderly person should be contacted by her physician or responsible care assistant (delegation of physician to the assistants). Furthermore, based on history, the same physician should be assigned to activities when the elderly person is in contact with the *alarm call center* staff (addressing the trust concern). After processing her history data and if everything is alright, her *last 7 days of vital signs* should be deleted. She should be informed that the processing has been performed and her health status is fine. She should be informed about the deletion of her history as well.

In step 6 and step 7 further iterative refinements of trustworthiness goals, and respectively in business processes, are performed. Gray-coloured elements (additional to the elicited trustworthiness goals) in Fig. 7 are the results of the refinement of the goal model. For instance, in order to satisfy *reliability and availability* the *redundant sensors for sending vital signs* are considered for providing the *vital signs* of the elderly person to the *alarm call center*. The task *Notify about usage and collection* is added to positively influence *privacy*. These refinements are further elaborated in business process models. Figure 9 shows further refinement of the trustworthiness requirements related to the *Notify elderly* activity which is related to the *Notify about usage and collection* from the goal model (cf. Fig. 7).

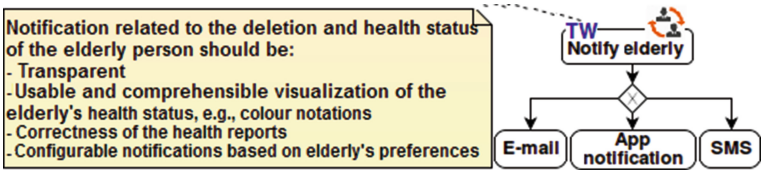


Fig. 9. Exemplary further refinement on business process model (within solution peak)

5 Related Work

The study of related work reveals some gaps in business process management with respect to trustworthiness. Plenty of works are done in security and to some extent in privacy. Short et al. [15] provided an approach for dealing with the inclusion of internal and/or external services in a business process that contains data handling policies. Wang et al. [21] developed a method to govern adaptive distributed business processes at run-time with an aspect-oriented programming approach. Policies can be specified for run-time governance, such as safety constraints and how the process should react if they are violated. Several works have been done to overcome the problem of considering qualities in resource assignment. Some meta-models like [10,20] and an expressive resource assignment language [2] have been developed. Between those, RALPH [2] provides a graphical

representation of the resource selection conditions and assignments. RALPH has formal semantics, which makes it appropriate for automated resource analysis in business process models. Stepien et al. [16] present the user interfaces that users can use to define conditions themselves. The main gap is addressing a broad spectrum of qualities which contribute to trustworthiness and the necessity of defining conditions on resources and activities in business process with respect to trustworthiness. The resource patterns provided by Russell et al. [14] are used to support expressing criteria in resource allocation.

Business Activities is a role-based access control extension of UML activity diagrams [17] to define the separation of duties and binding of duties between the activities of a process. Wolter et al. [22] developed a model-driven business process security requirement specification which introduces security annotations into business process definition models for expressing security requirements for tasks. However, current state of the art in this field neglects to consider trustworthiness as criteria for the resources and business process management.

6 Conclusions and Future Work

Managing business processes respecting trustworthiness requirements remains an ongoing challenge in service-oriented computing and cloud computing research. This paper discussed trust issues in the context of business process management using BPMN and i*. We provide an integration of subjective trust concerns into goal and process models. Our framework supports the analysis of a business process from activity, resource, and data object perspectives with respect to trustworthiness. To the best of our knowledge, we propose a novel contribution on user-centered identification of trust concerns and elicitation of trustworthiness requirements and thereafter integrating trustworthiness properties in business process design. Furthermore, our contribution includes a preparation for verification that satisfies trustworthiness constraints over resource allocation and activities executions.

We propose a method to identify the resources and activities that are trustworthiness-related. Then, we specify the trustworthiness requirements on those resources and activities in business processes with regard to trustworthiness goals from goal models. The proposed method needs a full integration to a business process modeling or management application. Furthermore, our framework supports the business process life-cycle with respect to trustworthiness.

This is a work-in-progress paper. The main ideas and findings will be further investigated and evaluated based on the presented example in Sect. 4. This will lead to the establishment of patterns and metrics for trustworthiness. To reduce the process designer's effort, we plan developing a set of patterns for easing trustworthiness requirement specifications. Our future research will focus on three important issues: (1) understand how the trustworthiness attributes actually influence trust. (2) how to identify interdependencies between different attributes of different parties involved in the business process, and how to consequently define a set of trustworthiness properties for process elements

and resources. (3) investigate existing risk assessment methodologies on the business process level, and show how they can support business process design and definition in building trustworthiness into processes in the whole life-cycle of business process management. We will improve our understanding and encourage the utilization of our framework and method by being perceived as useful, easy to use, easy to learn, compatible, and highly valued by practitioners.

References

1. Avancha, S., Baxi, A., Kotz, D.: Privacy in mobile technology for personal healthcare. ACM Comput. Surv. **45**(1), 1–54 (2012)
2. Cabanillas, C., Knuplesch, D., Resinas, M., Reichert, M., Mendling, J., Ruiz-Cortés, A.: RALph: a graphical notation for resource assignments in business processes. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) CAiSE 2015. LNCS, vol. 9097, pp. 53–68. Springer, Heidelberg (2015)
3. Di Cerbo, F., Gol Mohammadi, N., Paulus, S.: Evidence-based trustworthiness of internet-based services through controlled software development. In: Cleary, F., et al. (eds.) CSP Forum 2015. CCIS, vol. 530, pp. 91–102. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25360-2_8
4. Mohammadi, N.G., Bandyszak, T., Goldsteen, A., Kalogiros, C., Weyer, T., Moffie, M., Nasser, B.I., Surridge, M.: Combining risk-management and computational approaches for trustworthiness evaluation of socio-technical systems. In: Proceedings of the CAiSE Forum, pp. 237–244 (2015)
5. Mohammadi, N.G., Bandyszak, T., Kalogiros, C., Kanakakis, M.: A framework for evaluating the end-to-end trustworthiness. In: Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom) (2015)
6. Mohammadi, N.G., Bandyszak, T., Paulus, S., Meland, P.H., Weyer, T., Pohl, K.: Extending software development methodologies to support trustworthiness-by-design. In: Proceedings of the CAiSE Forum, pp. 213–220 (2015)
7. Mohammadi, N.G., Heisel, M.: Enhancing business process models with trustworthiness requirements, accepted. In: 10th IFIP WG 11.11 International Conference on Trust Management (2016)
8. Mohammadi, N.G., Heisel, M.: Patterns for identification of trust concerns and specification of trustworthiness requirements, accepted in the progress of publication (2016)
9. Mohammadi, N.G., Paulus, S., Bishr, M., Metzger, A., Könnecke, H., Hartenstein, S., Weyer, T., Pohl, K.: Trustworthiness attributes and metrics for engineering trusted internet-based software systems. In: Helfert, M., Desprez, F., Ferguson, D., Leymann, F. (eds.) CLOSER 2013. CCIS, vol. 453, pp. 19–35. Springer, Heidelberg (2014)
10. Koschmider, A., Yingbo, L., Schuster, T.: Role assignment in business process models. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 37–49. Springer, Heidelberg (2012)
11. Mei, H., Huang, G., Xie, T.: Internetware: a software paradigm for internet computing. Computer **45**(6), 26–31 (2012)
12. Nuseibeh, B.: Weaving together requirements and architectures. Computer **3**, 115–119 (2001)

13. OMG: Business Process Model and Notation (BPMN) version 2.0. Technical report (2011)
14. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: identification, representation and tool support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
15. Short, S., Kaluvuri, S.P.: A data-centric approach for privacy-aware business process enablement. In: van Sinderen, M., Johnson, P. (eds.) IWEI 2011. LNBIP, vol. 76, pp. 191–203. Springer, Heidelberg (2011)
16. Stepien, B., Felty, A., Matwin, S.: A non-technical user-oriented display notation for XACML conditions. In: Babin, G., Kropf, P., Weiss, M. (eds.) E-Technologies: Innovation in an Open World. LNBIP, vol. 26, pp. 53–64. Springer, Heidelberg (2009)
17. Strembeck, M., Mendling, J.: Modeling process-related RBAC models with extended UML activity models. Inf. Softw. Technol. **53**(5), 456–483 (2011)
18. Stroppi, L.J.R., Chiotti, O., Villarreal, P.D.: Extending BPMN 2.0: method and tool support. In: Dijkman, R., Hofstetter, J., Koehler, J. (eds.) BPMN 2011. LNBIP, vol. 95, pp. 59–73. Springer, Heidelberg (2011)
19. Sztompka, P.: Trust: A Sociological Theory. Cambridge University Press, Cambridge (2000)
20. van der Aalst, W.M.P., Kumar, A.: A reference model for team-enabled workflow management systems. Data Knowl. Eng. **38**(3), 335–363 (2001)
21. Wang, M., Bandara, K., Pahl, C.: Process as a service distributed multi-tenant policy-based process runtime governance. In: IEEE International Conference on Services Computing (SCC), pp. 578–585 (2010)
22. Wolter, C., Menzel, M., Schaad, A., Miseldine, P., Meinel, C.: Model-driven business process security requirement specification. J. Syst. Archit. Spec. Issue Secure SOA **55**(4), 211–223 (2009)
23. Yu, E.S.K.: Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, pp. 226–235 (1997)

Automatic Enforcement of Security Properties

Jose-Miguel Horcas(✉), Mónica Pinto, and Lidia Fuentes

CAOSD Group, Universidad de Málaga, Andalucía Tech, Málaga, Spain
{horcas,pinto,lff}@lcc.uma.es

Abstract. Ensuring the security requirements of an application is not a straightforward task. Security properties (e.g., confidentiality, anonymity) need to be satisfied in different ways in different parts of the same application. Software architects are usually required to manually define security components and their dependencies with the base application, customize them to the application's requirements, identify the points where security is incorporated, and verify that the selected places are correct. The last two steps are especially complex and error-prone. In our approach, we aim to provide a solution that helps software architects to identify the correct places to incorporate the security functionality and to verify the correctness of the composed application architecture. This is achieved by identifying a set of general structural patterns for incorporating security into the application architecture, and by providing a model-driven SPL solution to customize these patterns to each application's requirements.

Keywords: Encryption · Security pattern · Software architecture · SPL

1 Introduction

It is well known that the development of applications that ensure their security requirements is not a straightforward task [1–3]. This is because security properties (e.g., confidentiality, authentication, etc.) need to be satisfied in different ways in different parts of the same application. Even the same security property usually needs to be satisfied differently in multiple parts of the same application. For instance, in order to preserve confidentiality the application's sensitive data has to be encrypted. But, where and how does the data need to be encrypted?

In our example, confidentiality is guaranteed by adding the encryption behavior in different places of the application. For instance, during the interaction of two components that exchange sensitive data, it could be added at the place where the data is encrypted, and the place where the same data is to be decrypted. However, encryption could be applied differently to different kinds of interactions. For example, only remote interactions, involving components that are deployed in different hosts, may be required to encrypt sensitive information. Sensitive data could be encrypted for all the interactions managing sensitive information, independently from the location of components. Or, encryption is required for all interactions managing sensitive data but providing different

S. Katsikas et al. (Eds.): TrustBus 2016, LNCS 9830, pp. 19–31, 2016.
DOI: 10.1007/978-3-319-44341-6_2

security levels — i.e., using different encryption algorithms, depending on the local or remote location of the components. Moreover, guaranteeing the secure storage of the information requires the data to be encrypted before storing it and decrypted after retrieving it. More variability is introduced if we consider that the sensitive data have to circulate securely — i.e., encrypted, through different components of the application. This means that the component where the data is encrypted and the component where the data is decrypted do not directly interact with each other, as there are third components between them. Finally, different kinds of sensitive data may require different levels of security, requiring the use of different encryption algorithms (e.g., RSA, AES,...). Thus, it is not trivial for software architects to correctly answer the where-and-how question.

A first step toward mitigating this problem is applying the separation of concerns principle to model the variability of security from the early stages of the software's development. Thus, security concerns are modeled separately from the base applications and later customized to the application's requirements and composed at particular points of the application model [4,5]. It has been demonstrated that this approach has many advantages [4–6], such as high reusability, low coupled components and high cohesive software architectures. Moreover, security can be more easily customized to the application's requirements. Following this approach, our previous work [5] provided support to: (1) model the variability of security independently from the application, (2) instantiate the security model according to the requirements of a particular application, and (3) compose the customized security model with the application model.

However, in order to compose the customized security model and the application model, first, the join points — i.e., points in the application model where the elements of the security model have to be injected/composed, have to be identified. In our previous work, the join points were identified completely manually. This resulted in a lack of support to guarantee the required security level, since analyzing whether the security components had been introduced in all and the correct places was not possible. Other similar approaches [4,6,7] have the same limitation. Thus, the benefits of reusing the security models are lost. In this paper, we improve upon our previous approach by providing support to ensure that security is correctly incorporated in the applications. This is achieved by: (1) automatically identifying the places in the software architecture where a particular security functionality has to be incorporated, and (2) checking whether the security functionality was incorporated correctly to a software architecture.

In order to automate the identification of the join points we need to understand that security models are not completely oblivious to the application models, and thus some dependencies between them need to be taken into account during both the security modeling and during the incorporation of the security functionality inside the application. Without this, the automatic identification of the join points is impossible. Concretely, as part of the variability of the security properties, we need to model the variability of the different structural and behavioral patterns (e.g., the remote/local direct/indirect interactions, data storage, etc. in the case of encryption) previously discussed. The main reason is that the identification of the join points and the definition of the composition

rules largely depend on these patterns. Moreover, formally defining these patterns and the composition rules based on them will provide our approach with the support that is required to verify the correct deployment of security.

In this paper we focus on confidentiality, although our ultimate goal is to identify a set of general structural and behavioral patterns to incorporate many other security properties into an application's architecture, and to customize these patterns to each application's requirements. We use a Software Product Line (SPL) [8] to specify the variability of the composition patterns, and model-to-model (M2M) transformations to identify the join points from the patterns and to guarantee that the final architecture satisfies the security requirements.

The paper is structured as follows. Section 2 motivates our work with a case study. Section 3 presents our SPL to model and instantiate the variability of the security patterns. Section 4 explains the automatic identification of join points from the patterns. Section 5 qualitatively evaluates our approach. Section 6 discusses related work and Sect. 7 sets out our conclusions and future work.

2 Motivating Case Study

Our case study is an electronic payment (e-payment) application for making payments for different services (taxi, restaurants, donations,...) and chasing up receipts. This kind of application requires strong security requirements such as preserving confidentiality of the user's information, integrity of the data, and access control, among others. In this paper, we focus only on confidentiality.

Figure 1 shows a simplified UML software architecture with the basic functionality of the application. The `PaymentApp` component allows users to make payments for a specific service, and request the proof of payments by using the `EPayment` interface. The customer's information (e.g., payment card details) is stored on the user's device (`CustomerProfileManager` component). Additionally, this information can be synchronized between different user devices (`SynchronizationData` component). The server manages the payments through the `EPaymentServer` component that uses the `ServiceDomainResolution` and `BankTransaction` components to identify the service's provider information and to complete the transactions with the banks, respectively. The server also tracks a history of the users' transactions (`TransactionsHistory` component).

Apart from this basic functionality shown in Fig. 1, it is of paramount importance to guarantee the following security requirements, among others:

- *Req. 1: Confidentiality.* Sensitive information (i.e., payment card data) exchanged between the client and the server hosts must be encrypted (e.g., using the RSA algorithm). This means that it is required to encrypt the payment information (information of type `PaymentMethod` or the parameter `payInfo` in Fig. 1) when: (1) a payment is made — i.e., the `PaymentApp` client component calls the `pay` method of the `EPayment` interface; and when (2) the client synchronizes the list of payment methods — i.e., the component `CustomerProfileManager` calls the `synchronize` method of the `SynchInfo` interface. Then, the information has to be decrypted when the

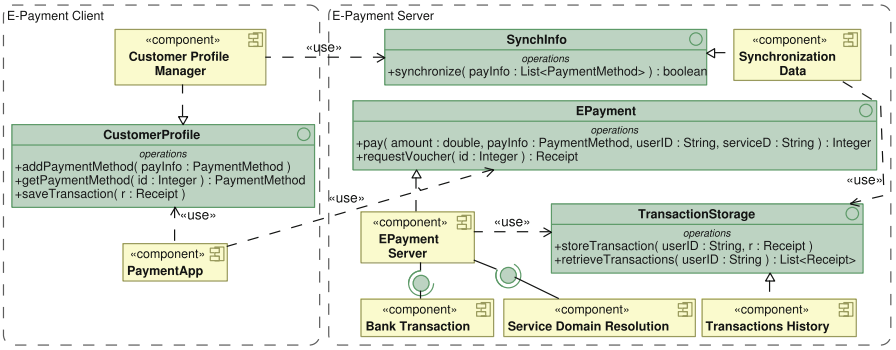


Fig. 1. Software architecture of the e-payment application.

server receives it, in both `EPaymentServer` and `SynchronizationData`. Following our approach the software architect only needs to indicate that the payment card data is the sensitive data and then our automatic join point identification approach indicates those join points where encryption and decryption need to be incorporated in order to ensure the confidentiality of the sensitive data.

- *Req. 2: Confidentiality.* All information exchanged between the `EPaymentServer` and `BankTransaction` components inside the server must also be encrypted, regardless of the type of information or the interface used. In this case, the software architect indicates that interactions between two specific components need to be encrypted and all the affected join points are automatically identified by our approach.
- *Req. 3: Confidentiality.* The payment card details are stored in the user's device using a different encryption algorithm from the one for communications (e.g., AES). Thus, another encryption algorithm is required to encrypt the payment methods information when they are stored/retrieved in the user's device. In this case, both encrypting and decrypting functionalities are required by the same component (`CustomerProfileManager`). Here, our approach inspect the application looking for a structural pattern that represents a data storage and the join points would be automatically detected.

3 Capturing the Security Variability

To accomplish the automatic identification of the joint points, we identify a set of structural patterns that specifies the relationships with the application. The variability of the patters is modelled in an SPL, together with the variability of the security functionality. Then, the software architect instantiates the patterns and the security functionality according to the application's requirements.

A security pattern describes a particular, recurring security problem (e.g., applying encryption) that arises in specific contexts, and presents a well-proven generic solution for it [9]. In the case of the confidentiality property, we need

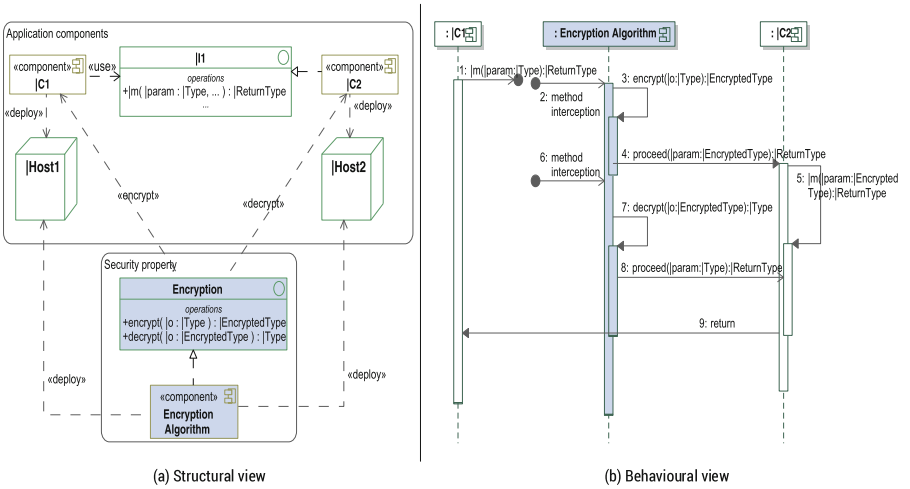


Fig. 2. Encryption pattern for secure communications.

a set of patterns that allows to apply the encryption functionality to different parts of the application such as remote/local direct/indirect interactions, and data storage. Encryption is usually defined as a component that provides two main functionalities: `encrypt` and `decrypt` (see Fig. 2), which normally intercept (*crosscut*) the application functionality at specific points. For example, the pattern to apply encryption in a secure communication is shown in Fig. 2 and states that the `encrypt` method intercepts a "sender component" (`|C1`) in order to encrypt the message information (`|param`) before sending it (i.e., calling the method `|m`), while the `decrypt` method crosscuts a "receiving component" (`|C2`) to decrypt the message information after receiving it. Figure 2 represents a parameterizable structural (a) and behavioral (b) view of this pattern for encrypted communications. The top of Fig. 2(a) shows the dependencies of the pattern with the architectural elements of the application architecture (`Application components`). The bottom of Fig. 2(a) shows the encryption component (`EncryptionAlgorithm`) with the provided functionality (`Encryption` interface). The pattern captures the information that is required to incorporate encryption into the interaction between two components. Throughout this paper, we only use the structural view for the sake of simplicity, but patterns can be complemented with additional views as shown in Fig. 2(b).

The partial or total instantiation of the parameters of this pattern, with information obtained from the application's requirements, allows correctly applying encryption for straightforward communications — i.e., applying encryption to two communicating components that use a common interface. However, this pattern does not capture all the situations in which encryption may have to be incorporated. For instance, it does not allow applying encryption in situations that do not involve a communication, such as storing encrypted data in a device, or applying encryption to interactions between two non-adjacent components.

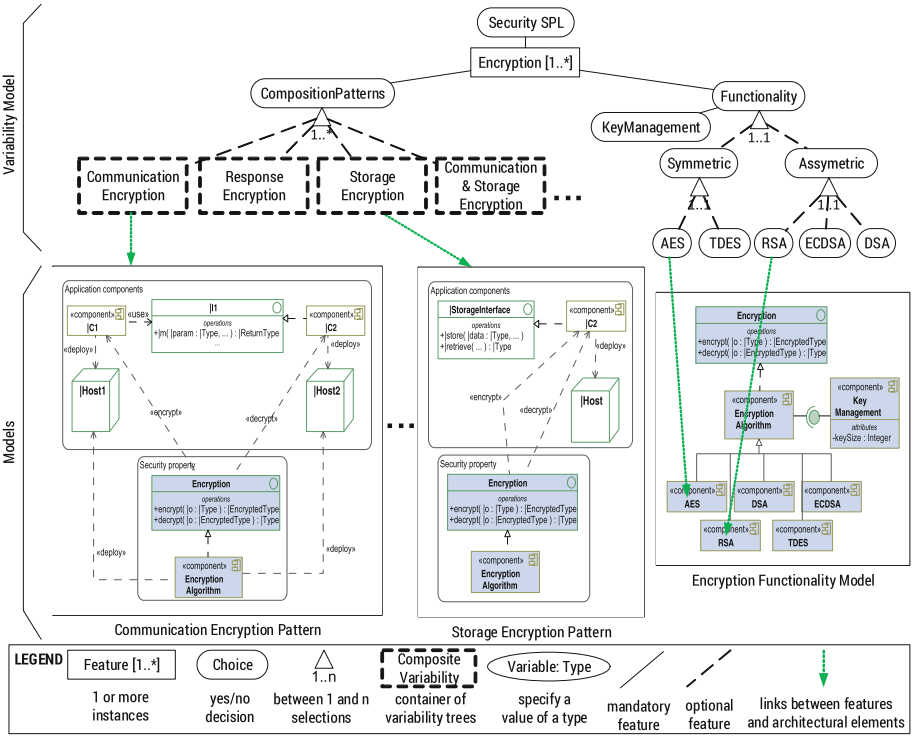


Fig. 3. Variability model for the encryption patterns and encryption functionality.

Thus, as it is impossible for only one pattern to cover all kinds of interactions with the application, we propose modeling the variability of the security patterns following a Software Product Line (SPL) [8]. Concretely, we extend the variability model modeling the security functionality in our previous work to enhance it with the variability of the security patterns. An SPL[1] allows us to specify the commonalities and variabilities of a product and then generate specific configurations of the product according to different requirements.

Figure 3 shows a variability model that specifies, using *features* in an abstract level (top of Fig. 3), the variability of the encryption patterns (left of the figure) and the variability of the encryption functionality (right of the figure). Then, specific models (e.g., the software architecture of encryption, the structural/behavioral patterns for encryption) are linked to the features of the abstract tree. Note that using existing tools for SPL (e.g., CVL [10], SPLOT [11]), the architectural models in the bottom of Fig. 3 will be automatically instantiated according to the features selected from the abstract tree. Concretely, the bottom of Fig. 3 shows two parameterizable encryption patterns and the model of the encryption functionality (`Encryption Functionality Model`), including all the

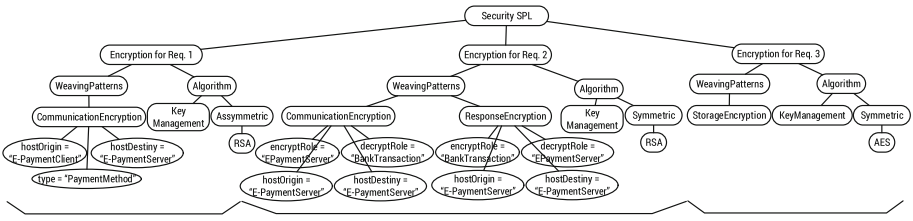
[1] http://www.sei.cmu.edu/productlines/.

variable architectural elements. Notice that security properties are usually modeled by much more complex architectures, though in this example we only represent the encryption algorithms for the sake of simplicity. For the `Encryption Functionality Model`, a selection of a particular feature in the tree selects the encryption algorithm that will be used. The multiplicity feature (`Encryption [1..*]` in Fig. 3) indicates that both the algorithms and the patterns can be instantiated multiple times in order to use different algorithms and patterns.

3.1 Resolving the Variability of the Application

Once all the variability of the security functionality and the patterns has been defined in the SPL (only once) by the domain experts, the software architect can use our approach to generate different configurations of the patterns and the security functionality according to each application's requirements.

A complete configuration of the variability model from requirements `Req. 1`, `2`, and `3` of our case study is shown in Fig. 4. There are three instances of the encryption feature: one for each requirement. The first instance (`Encryption for Req. 1`) is configured with the `RSA` algorithm, and uses the `Communication-Encryption` pattern instantiated as shown in Fig. 5(a), with the goal of encrypting the sensitive information exchanged between the client and the server host. The second instance (`Encryption for Req. 2`) is also configured with the `RSA` algorithm, but uses two patterns (`CommunicationEncryption` and `Response-Encryption`) to apply encryption in both directions of the communications between the components `EPaymentServer` and `BankTransaction` of the same host (`E-PaymentServer`). Since all information exchanged between these two components is required to be encrypted, no information regarding the type of the data, nor the interface, methods, etc. is provided by the software architect. Finally, the third instance (`Encryption for Req. 3`) is configured with the `AES` algorithm, and uses the `StorageEncryption` pattern in order to storage the payment information in a secure way. In this case, the software architect has not instantiated any parameter of the pattern, as shown in Fig. 5(b). This could



Req. 1. Sensitive information (`PaymentMethod`) exchanged between the client (`E-PaymentClient`) and the server (`E-PaymentServer`) hosts must be encrypted using the `RSA` algorithm.

Req. 2. All information exchanged between the `EPaymentServer` and `BankTransaction` components inside the server (`E-PaymentServer`) host must be encrypted using the `RSA` algorithm.

Req. 3. The payment card details are stored in the user's device using the `AES` algorithm.

Fig. 4. Instance of the variability model for the encryption functionality and patterns.

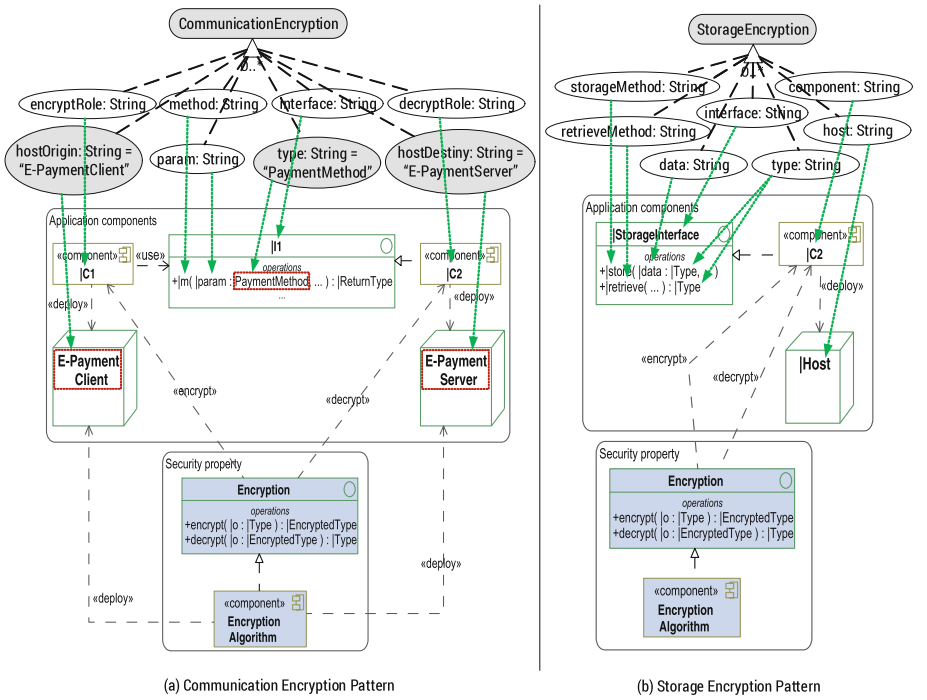


Fig. 5. Instances of the encryption patterns for (a) *Req 1.* and (b) *Req. 3.*

occur when the requirements do not provide enough information, the software architect does not know how to interpret the requirements, or does not have the required knowledge to instantiate the pattern. In this case our approach identifies a larger set of join points and the software architect has to manually select the correct ones. At least it knows all the matching points that need to be considered.

The encryption patterns show the parameters that can be customized according the application's requirements. The patterns for applying encryption to the communications (`Communication Encryption Pattern`) and to the storage of information (`Storage Encryption Pattern`) are described in more details in Fig. 5(a) and (b), respectively. Providing a value in the feature tree means that this element in the pattern will be instantiated with the provided value. For instance, to satisfy *Req. 1*, the software architect has instantiated the `Communication Encryption Pattern` of Fig. 5 (a) with the following parameters: the data type of the information to be encrypted (i.e., the `PaymentMethod` type), and the identifiers of the client and the server hosts (`E-PaymentClient` and `E-PaymentServer`). The following section explains how we correctly identify the join points from the previously customized patterns in our approach.

4 Supporting the Composition Process

Once the variability model has been instantiated, now the security and the application models are composed. To achieve this, the composition patterns customized in the previous step are mapped on structures in the application architecture by using M2M transformations. The mapping process can be used in two complementary ways: (1) guiding the software architects in selecting the correct join points, and (2) supporting them in verifying their choices of join points.

4.1 Automatically Identifying the Join Points

In order to identify the join points where the customized patterns have to be applied and guaranteeing that the final architecture satisfies the security requirements, the model transformations can be treated as a separate previous step to the composition process. Before the composition process, checking each instantiated pattern with our e-payment application architecture (Fig. 1) finds all possible matchings where the pattern can be applied (Fig. 6). The number of identified matchings directly depends on the number of parameters for which a specific value was provided during the instantiation of the variability model.

For instance, the first instantiated pattern (`Encryption for Req. 1`) matches the application model in the join points `Req. 1. Matching 1` and `Req. 1. Matching 2` in Fig. 6. The software architect provided just the data type to be encrypted (`PaymentMethod`) and the hosts' information, while the concrete components where the encrypt and decrypt methods will be composed has

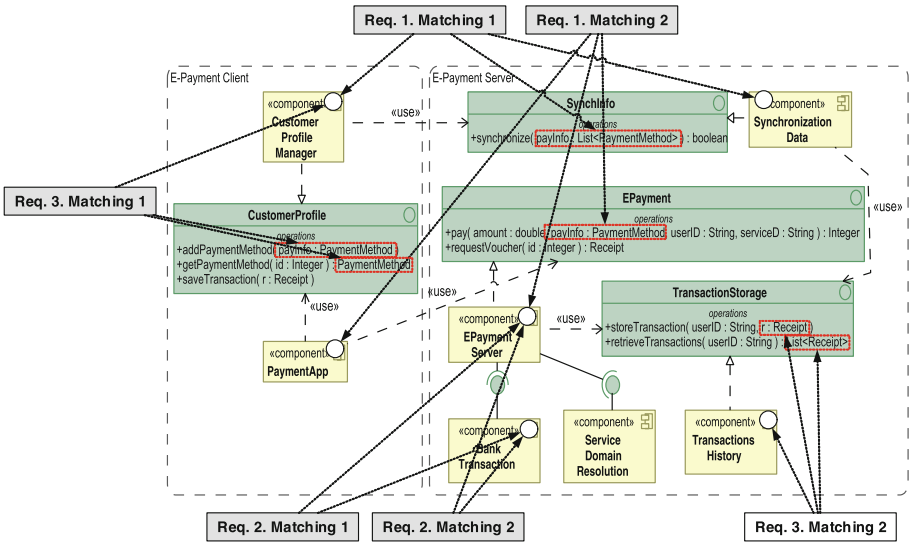


Fig. 6. Matchings for the encryption patterns in the e-payment architecture.

been automatically identified. To satisfy *Req. 2*, two patterns were instantiated: `CommunicationEncryption` and `ResponseEncryption`. Each of them matches the application architecture in `Req. 2. Matching 1` and `Req. 2. Matching 2`, respectively. In this case, the identifiers of the two communicating components were directly provided: the `PaymentServer` and the `BankTransaction` components. All information exchanged between these two components will be encrypted before sending and decrypted after being received. Finally, for *Req. 3*, the `StorageEncryption` pattern was instantiated without specifying any parameters. This implies that there will be multiple matchings for this pattern, as `Req. 3. Matching 1` and `Req. 3. Matching 2` in Fig. 6. Both matchings are correct for this pattern. However, *Req. 3* only specifies that the payment card information that is stored in the user's device need be encrypted (`Req. 3. Matching 1`) and, thus, the information about the receipts of the transactions (`Req. 3. Matching 2`) does not need to be encrypted. In such a case, we give the software architect the opportunity to make an explicit choice of the matching, or to instantiate the pattern again by providing more specific information such as the type of data to be encrypted (e.g., the `PaymentMethod` in this case).

4.2 Verifying the Security Requirements

To demonstrate that the security functionality has been applied in the correct way and places, the M2M transformations can be applied to a model where the security model has already been composed with the application model. For instance, when the join points were manually identified. The same instance of the variability model shown in Figs. 4 and 5 can now be used to verify that the security property was correctly added to all the matchings of the final application model (the base application model composed with the security model). In our case study, a software architect could verify whether the final application model includes encryption in all the matchings shown in Fig. 6. Comparing the final application model and the matchings, the software architect may realise that encryption was not added to some of the identified matchings. This supposes a hole in the security of the application, but it can be easily resolved according to the information provided automatically by our approach. Thus, the tool supporting the instantiation of the patterns can also be used to check that the software architect has applied them in all the correct places.

5 Evaluation Results and Discussion

We have tested the validity of our approach by implementing the patterns and M2M transformations using the Henshin transformation language [12].[2] We have used two case studies: the e-payment application used throughout this paper, and an electronic voting application.[3] In this section we qualitatively argue the correctness, extendibility, and reusability of our approach.

[2] They are available at http://150.214.108.91/code/interfacesfqa/tree/master.
[3] http://inter-trust.eu/.

Correctness. The correctness of our approach depends on the correctness of the specification of the patterns and on the implementation of the M2M transformations. The patterns and M2M transformations are formally modeled conforming to a specific metamodel, so if the domain experts do their job correctly, the identification and checking of the join points will also be correct. Moreover, separately modeling the security functionality and the base application considerably facilitates the verification of the security properties of an application since a security expert can rely on the automatic output provided by applying the patterns, instead of manually checking all the modules in the base application to ensure that all security requirements have been correctly enforced. Finally, our approach is able to ensure the level of security required by an application even when the software architect is not completely aware of the elements in the application architecture that are affected by the security requirements, but is able to indicate at least, the structural patterns that are affected by security (e.g., encrypt the communications between components, or encrypt the data store in a data storage). In this case, our approach identifies a larger set of join points because most of the pattern's parameters are not specified. We can ensure that all of them are correct. The software architect then has two options: (1) add encryption to all the identified join points. This will guarantee that the security of the application is ensured, or (2) manually select a subset of them. In this case, our approach is not responsible for the security gaps that may be introduced.

Extendibility. In this paper we have focused on the confidentiality property and have shown in the SPL only the variability of the encryption algorithms. However, the SPL can be easily extended to cover more security properties such as authentication, integrity, anonymity, etc. Moreover, the variability model can consider any variable security functionality such as the management of the keys for encryption or the passwords for the authentication concern, not just the variation between algorithms. Note that although the intricacy of this approach may seem inadequate for only adding encryption to an application, our final goal is much more ambitious as this work is part of an approach to separately modeling the variability of quality attributes [5]. Concretely, we have an SPL modeling the variability of several quality attributes, not only security (e.g., contextual help, persistence) and the approach presented in this paper is applicable to all of them.

Reusability. Our approach improves the reusability of the security concerns by modeling the security functionalities separately from the core functionality of the application, from early stages. This reduces the coupling and increases the cohesion of software architectures. Also, thanks to the combined use of the separation of concerns and the SPLs, we can reuse the same security functionality and patterns with different applications.

6 Related Work

Security is usually achieved in several ways, but most of the approaches present security as a set of non-functional properties [6,9,13,14], instead of focusing on

the functional part of the security concerns as we have done for confidentiality in this paper. For instance, in [6], the authors present a systematic approach for weaving non-functional requirements into software architecture using architectural tactics similar to our composing patterns. However, we consider security as extra-functionality that needs to be present as functional components inside the application architecture to satisfy the requirements. So, we focus on the identification of the correct places where security must be incorporated, instead of providing the systematic steps to perform the composition of the patterns, as we also did in previous work on composing security functionalities [5,15].

Cuevas et al. [7] also describe a generic solution for non-security experts using security patterns. The solution captures a security pattern that provides access control to sensor data based on light-weight encryption and grant provision. No means of how and where applying encryption functionality to the software architecture is described. Only the properties and functionalities that are common to all implementations of the encryption-based access control are captured using the security patterns, and thus, the customization of the patterns is too limited because of the lack of variability. QADA [16] is a specific method for designing SPL architectures by transforming systematic functionality into software architectures, but this proposal does not explicitly take into account the security requirements, so the semantic correctness of the final architecture cannot be checked, in order to assure the quality of the system.

Another approach that separately models the security functionality from the base application is CORE (Concern-Oriented REuse) [4]. Nevertheless, as the other existing work [5–7,15], they do not provide mechanisms to guarantee that security is deployed in all and correct places of the application architecture.

7 Conclusions and Future Work

In this paper we have presented an approach towards the automation of the composition process between application and security models. Specifically, the approach consists in modeling the variability of a set of patterns to incorporate the security functionality in the correct places of the application architecture, according to its requirements. This means that we provide the software architect with support for automatically identifying the join points where the security functionality has to be incorporated. So, instead of manually identifying the join points, as existing approaches propose, the system offers the software architect a set of join points. It also provides support to verify the correctness of a composed application architecture, so that the requirements of the system can be assured. As future work we plan to improve our approach by defining the patterns in terms of a security conceptual model that will allow the join points to be selected based on semantic instead of just syntactic information [17].

Acknowledgment. This work is supported by the project Magic P12-TIC1814 and by the project HADAS TIN2015-64841-R (co-financed by FEDER funds).

References

1. Preda, S., Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J., Toutain, L.: Model-driven security policy deployment: property oriented approach. In: Massacci, F., Wallach, D., Zannone, N. (eds.) ESSoS 2010. LNCS, vol. 5965, pp. 123–139. Springer, Heidelberg (2010)
2. Ayed, S., Idrees, M.S., Cuppens-Boulahia, N., Cuppens, F., Pinto, M., Fuentes, L.: Security aspects: a framework for enforcement of security policies using AOP. In: SITIS, pp. 301–308 (2013)
3. Mouelhi, T., Fleurey, F., Baudry, B., Le Traon, Y.: A model-based framework for security policy specification, deployment and testing. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 537–552. Springer, Heidelberg (2008)
4. Alam, O., Kienzle, J., Mussbacher, G.: Concern-oriented software design. In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P. (eds.) MODELS 2013. LNCS, vol. 8107, pp. 604–621. Springer, Heidelberg (2013)
5. Horcas, J.M., Pinto, M., Fuentes, L.: An automatic process for weaving functional quality attributes using a software product line approach. J. Syst. Softw. **112**, 78–95 (2016)
6. Kim, S., Kim, D.K., Lu, L., Park, S.: Quality-driven architecture development using architectural tactics. J. Syst. Softw. **82**(8), 1211–1231 (2009)
7. Cuevas, A., Khoury, P.E., Gomez, L., Laube, A.: Security patterns for capturing encryption-based access control to sensor data. In: SECURWARE, pp. 62–67 (2008)
8. Pohl, K., Böckle, G., van der Linden, F.J.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer, New York (2005)
9. Schumacher, M., Fernandez, E., Hybertson, D., Buschmann, F.: Security Patterns: Integrating Security and Systems Engineering. Wiley, Chichester (2005)
10. Haugen, Ø., Wasowski, A., Czarnecki, K.: CVL: common variability language. In: Software Product Line Conference, SPLC, vol. 2, pp. 266–267 (2012)
11. Mendonca, M., Branco, M., Cowan, D.: S.P.L.O.T.: software product lines online tools. In: Object Oriented Programming Systems Languages and Applications, OOPSLA, pp. 761–762. ACM (2009)
12. Arendt, T., Biermann, E., Jurack, S., Krause, C., Taentzer, G.: Henshin: advanced concepts and tools for in-place EMF model transformations. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010, Part I. LNCS, vol. 6394, pp. 121–135. Springer, Heidelberg (2010)
13. Yu, H., Liu, D., He, X., Yang, L., Gao, S.: Secure software architectures design by aspect orientation. In: ICECCS, pp. 47–55 (2005)
14. Hafiz, M., Adamczyk, P., Johnson, R.E.: Organizing security patterns. IEEE Softw. **24**(4), 52–60 (2007)
15. Horcas, J.M., Pinto, M., Fuentes, L.: An aspect-oriented model transformation to weave security using CVL. In: MODELSWARD, pp. 138–147 (2014)
16. Matinlassi, M., Niemelä, E., Dobrica, L.: Quality-driven Architecture Design and Quality Analysis Method: A Revolutionary Initiation Approach to a Product Line Architecture (2002)
17. Pires, P.F., Delicato, F.C., Pinto, M., Fuentes, L., Marinho, É.: Software evolution in AOSD: a MDA-based approach. In: CBSE, pp. 193–198 (2011)

Security and Privacy in Cloud Computing

Towards a Model-Based Framework for Forensic-Enabled Cloud Information Systems

Stavros Simou[1(✉)], Christos Kalloniatis[1], Haralambos Mouratidis[2], and Stefanos Gritzalis[3]

[1] Cultural Informatics Laboratory, Department of Cultural Technology and Communication, University of the Aegean, University Hill, GR 81100 Mytilene, Greece
{SSimou,chkallon}@aegean.gr

[2] School of Computing, Engineering and Mathematics, University of Brighton, Watts Building, Lewes Road, Brighton, BN2 4GJ, UK
H.Mouratidis@brighton.ac.uk

[3] Information and Communication Systems Security Laboratory, Department of Information and Communications Systems Engineering, University of the Aegean, GR 83200 Samos, Greece
sgritz@aegean.gr

Abstract. One of the most important challenges for software engineers is the design and implementation of trustworthy cloud services. Information system designers face an important issue, the design of cloud forensic-enabled systems that could assist investigators solving cloud-based cyber-crimes. Although digital forensics assists on this direction, limited evidence of cloud-based forensic approaches exist. These approaches don't support information systems developers as they focus on the investigation only and also they don't support modelling potential cases of forensics investigations. This paper aims to fill this gap by introducing a modelling language, presented in terms of a meta-model. Since most respective efforts focus on the investigation part a thorough analysis and a suggestion of a generic cloud forensic process is included as the main input for designing the proposed language.

Keywords: Cloud forensic framework · Cloud forensic process · Cloud forensic meta-model · Digital forensic models · Cloud forensics

1 Introduction

Over the past years, cloud computing altered the way services are provided to users and organizations. This is due to its high scalability and pay-per-use utility model. In January 2015, RightScale [1] conducted a Cloud Survey of the latest cloud computing trends and revealed that cloud computing adoption continues to be a given, with 93 percent of IT professionals reporting that they are adopting cloud services. International Data Corporation [2] predictions for 2015, presented by Frank Gens, mention that cloud services will remain a hotbed of activity in 2015 with $118 billion in spending on the greater cloud ecosystem, growing to over $200 billion by 2018. Increasing interest in use of cloud computing automatically means increasing interest on cyberattacks by

S. Katsikas et al. (Eds.): TrustBus 2016, LNCS 9830, pp. 35–47, 2016.
DOI: 10.1007/978-3-319-44341-6_3

perpetrators. According to PricewaterhouseCoopers [3], in 2015, there was an increase of 38 % in detected information security incidents from the previous year and an increase of 56 % in theft of hard intellectual property. A survey in December 2014 by Open Data Center Analysis [4] revealed that the primary concern that slows cloud adoption remains security, with more that 70 % of respondents supporting that. This creates the need for information system engineers to design forensic-enabled services in order to resolve cloud incidents (cyber-attacks) as fast and efficient as possible raising in parallel the trustworthiness of the services provided. Conducting an effective cloud forensic investigation requires, from an information systems development point of view, the support of the designers in identifying requirements that will assist developers to build forensic-enabled information system; this is an information system that its architecture supports forensic investigation. While there are several research works in the field of digital forensics, there are only few concerning cloud forensics. Moreover, the literature lacks work to support software engineers in identifying forensic-related requirements for information systems. For addressing the aforementioned gaps our work is concentrating on the establishment of a common modeling language presented in terms of a meta-model, which includes all necessary concepts for designing a cloud-forensic enabled system. One of the main prerequisites for designing the meta-model was the understanding of how cloud-forensic investigation is conducted. Since most of the research efforts are concentrated on the investigation part this paper proposes a generic cloud forensic investigation process in order to clarify all necessary activities required by the investigators for fulfilling their task. The understanding of this process as well as an extensive literature review on respective concepts and challenges for cloud forensics presented in our previous work [5–7] assisted on the design of the proposed modeling language.

The rest of the paper is organized as follows: Sect. 2 presents the cloud forensic investigation process. In Sect. 3 the proposed meta-model is presented. In Sect. 4 a running example for addressing the concepts of the meta-model is presented. Finally, Sect. 5, concludes the paper by raising future research on this innovative research field.

2 A Process for Cloud-Forensic Investigation

After a thorough analysis [5–7] of the respective literature this paper proposes a generic process for cloud-forensic investigation consisting of the following steps: Incident Confirmation, Incident Identification, Collection-Acquisition, Examination-Analysis and presentation. The proposed process is also illustrated in Fig. 2. Understanding the cloud forensic investigation process is of vital importance in order to identify the key factors that a modeling language aiming on modeling Cloud-Forensic enabled systems must address.

2.1 The Process

Incident Confirmation. The first stage is the confirmation of the incident. An incident may be detected by different sources such as an automated detection system,

administrator, external actors, or accidentally. In the confirmation stage the protective actors are made aware that an incident has detected and reported. According to Ciardhuain [8] the awareness stage need to be included because the events causing the investigation may influence the type of investigation required. The people responsible for the safety (protective actors) need to be informed about the malicious action and start searching the incident using all available resources to realize what it concerns. It can be a breach on confidential data, stolen information, a DDOS attack, trafficking illegal content, etc. Protective actors should be able to understand the nature of the incident and decide if they are willing to proceed with an investigation or not. Their decision involves different factors such as the criticality and severity of the incident, the infection (damage can cause), the cost and the availability on human resources. Kohn [9], states that "the detected incident should be confirmed by some other source before action is taken towards an incident response". Once the incident is confirmed, protective actors need to notify and inform all the stakeholders involved in the investigation. Application of warranty should be prepared and appropriate authorizations need to be obtained in order to grant permissions to different stages of the investigation. On the other hand, if the incident does not impose an immediate threat to organizations, or public security and it can be solved by the inside, then, the investigation is not initiated.

Incident Identification. The next step is to identify all relevant assets (software, hardware and data) that may contain potential evidence, to build a case. According to [10], identification is the "process involving the search for recognition and documentation of potential digital evidence". Protective actors need to determine the type of crime and what type of assets are used. Protective actors also need to identify the assets (potential evidence), the location of the incident, the malicious actor's resources and the cloud provider. An important concern is the trustworthiness of the involved CSP. As Zawoad [11] mentioned, most of the existing work on cloud forensics is taking a priori that CSPs are trusted entities and honest in a cloud investigation. "Trust must be managed through detailed Service Level Agreements (SLAs) with clear metrics and monitoring mechanisms and clear delineation of security mechanisms" [12]. Once the incident is confirmed, an investigation team should be formed consisting of people with special skills in cloud environments, such as legal advisors, experienced technicians and law officers. Warrant permissions to different stages of the investigation should be granted. All the actions taken should be recorded and documented. A proper documentation can be very helpful in the next stages of the investigation and in parallel it can maintain the chain of custody. Protective actors also need to consult previous cases and all the action plans performed during their training in order to prepare and deploy their strategy. Initial planning is based on respective older documentation and policies. Authorizations should be obtained to carry out the investigation and resources need to be identified. Resources include the personnel (actors) that will form the team to cope with the investigation, the methods and procedures that they will adopt and the tools they will use to identify the potential evidence. An actors list, assets list, system information report, time plan, acquisition plan and action plan (risk assessment plan) will be produced and recorded to maintain the chain of custody. Finally, the Service Level Agreement (SLA) between

CSPs and cloud consumer should be reviewed by the actors to understand technical and legal terms.

Collection – Acquisition. After identifying the assets and their location, the collection and acquisition process follows. The goal of this phase is to obtain the potential evidence. Depending on specific factors such as the kind of potential evidence, the criticality of the system or the legal requirements, the actors should decide what type of method must be used to extract them. In an ongoing cloud investigation, the impact for seizing hardware equipment cannot be measured; hence, in most of the cases, acquisition method should be used. [10], defines collection as the "process of gathering the physical items that contain potential digital evidence", meanwhile, acquisition is defined as the "process of creating a copy of data within a defined set". The methods of collecting data are either static or live. In the first case, the process is straightforward; seizing the items and removing them to a forensic lab for further examination. In the second case, the systems are running and the collection is performed on a system in running state. This involves an image or a snapshot acquisition that it can obtain useful information about registry entries, temporary files, memory, running processes, log entries, cache, etc. According to [13], "the copy created during acquisition can range from the forensic image of a hard drive to a copy of the contents of a server's memory to the logical contents of an individual user's email box". Pichan [14], states that for the cloud a series of snapshot images over a period of time should be taken in order to provide all the information regarding changes. During the collection-acquisition stage specific resources will be used. This involves well-trained personnel (internal or even external actors), special tools for cloud extraction data and up-to-date methodologies/processes such as protection mechanisms and action plans. Using the appropriate resources, protective actors aim to obtain both volatile and non-volatile potential evidence, in a forensically sound manner. The acquired assets should be securely stored for further analysis. The acquired evidence should be well-documented and checked for their integrity using hash methods and algorithms in order to discover any future alteration.

Examination – Analysis. Once the acquired data has been stored in a safe and secure storage a number of identical copies to the original data should be produce in order the protective actors to work with. This process involves two different sub-processes: evidence examination and evidence analysis. According to NIST [15], examination is defined as "the involvement of forensically processing large amounts of collected data using a combination of automated and manual methods to assess and extract data of particular interest, while preserving the integrity of the data" while analysis is defined as "the process to analyze the results of the examination, using legally justifiable methods and techniques, to derive useful information that addresses the questions that were the impetus for performing the collection and examination". In order to go into a forensic examination, protective actors should obtain a high level overview of the terrain and form a strategy; otherwise, delays might occur when unforeseen but preventable problems are encountered. This phase "is an important step for data collected from a cloud computing environment as the data is unlikely to be stored and collected in a form which permits immediate forensic analysis" [16]. Technician examiners should be

informed by the questions and priorities that protective actors developed during their initial planning [17]. On the other hand, examiners should review previously encountered cases and training plans to find patterns that can help reduce the time of the examination and develop their action plan. The enormous amount of data collected in the previous stage should be converted into manageable size and form for future analysis [18]. Due to the volume and complexity of data stored on digital devices, examiners should take decisions on what methods and tools should use in order to focus on the relevant data [19]. Examiners should search for timestamps, usernames and passwords, particular keywords using filters, etc. During analysis, actors should determine the significance of the data in order to transform them into evidence. Encrypted data should be processed and analyzed and the results will be used to reconstruct the timeline. Metadata from the examination phase will be analyzed and correlated to the potential evidence. Also, "metadata and other forms of audit data must be properly kept and made available when requested" [16]. The tools used will permit analysts to group related events into meta-events [15]. This process may perform several iterations to support the investigation depending on the evidence during the analysis phase. It could iterate back to the collection-acquisition or even identification process.

Presentation. The last stage is the presentation of the evidence selected during the investigation. [20], states that presentation process involves three steps in order to ensure a successful conclusion to the investigation; these are case preparation, case presentation and evidence preservation. Experts should be prepared to confront the jury who lacks knowledge of cloud computing and try to present the evidence collected in a language that anyone can understand. [15], uses the word reporting for this process and defines it as "the process of preparing and presenting the information resulting from the analysis phase". During presentation, the personnel responsible for presenting the respective report should be well prepared to explain in a logical and understandable way the preserved and documented evidence. The implemented reports along with the supporting materials concerning the chain of custody of the evidence should be submitted to the court of law. At the end of the trial, the evidence and the documentation should be carefully stored and secured in order to be used either in case of an appeal or for future purpose.

Concurrent Activities. Some activities are running in parallel with the aforementioned stages. These are the preservation of the evidence, documentation and preparation (training and planning). In a cloud environment, the challenge is how to preserve the data and then determining whether the existing approaches of measuring data integrity are applicable or not [21]. To ensure that the integrity of evidence and the chain of custody are maintained throughout the investigation, this activity should be running in parallel with all the stages of the aforementioned process. The same applies for the documentation activity. For conventional forensic process, [22] defines chain of custody as "a roadmap that shows how evidence was collected, analyzed and preserved in order to be presented as evidence in court". Braid [23], states, that the evidence must meet five criteria in order to be used and support a trial, these are: admissible, authentic, complete, reliable and believable. To preserve the integrity of the evidence, maintain

the authenticity and the chain of custody a number of requirements need to be produced, such as reports (handling, methodology, storage, etc.), lists (tools, actors, procedures, etc.) and logs (activity logs). Any change that will produce a different result should be recorded. According to Prayudi [24] protective actors are facing a serious problem in the chain of custody related to the documentation of the evidence. This is due to tremendous amount of data and the distributed cloud environment that require many different concepts and entities to handle the evidence. The main objective of the documentation is to keep the investigation proper documented in order to increase the probabilities of winning a case in a court of law. The main objective of preparation (training and planning) activity is to prepare and ensure that personnel, operations and infrastructures are able to support an investigation in case of an incident [25]. A well-organized preparation can improve the quality and availability of digital evidence collected and preserved, while minimizing cost and workload [26]. Training plans will be used as input in order to organize and prepare the resources of the investigation. SLAs are contracts providing information on how a cloud forensic investigation will be handled, usually signed between consumers and CSPs [27]. Well-written and robust SLAs should be considered in order to provide technical and legal details about the roles and responsibilities between the CSP and the cloud customer, security issues in a multi-jurisdictional and multi-tenant environment in terms of legal regulations, confidentiality of customer data, and privacy policies.

3 Meta-Model

In this section a revised meta-model from our previous work [7] is being introduced. The goal of the specific meta-model is to present all necessary concepts that will assist the designers in modeling all respective aspects when designing a cloud forensic enabled system/service.

The forensic investigation process is initiated whenever an incident occurs. Once the incident is brought to investigators' attention the forensic investigation process is initiated. Malicious actors are the ones introducing an incident and protective actors are people investigating it and trying to find a solution. On the other hand, whenever there is an attack there is always a target (victim). In cloud forensics, targets are usually individuals, organizations, companies, etc.

Malicious actors use Cloud Service Providers' services to launch their attacks hidden behind anonymity. CSPs major concern is to rent as many services to clients. So far we distinguished four different actors involved in a cloud forensic investigation: malicious actors, protective actors, cloud provider and the victim. An incident most of the times affects one target (i.e. user or machine) and in parallel introduces goals (to solve the incident, find perpetrators, etc.). All actors use resources (personnel, tools, trainings plans, methods, etc.) either to create the incident or to resolve it. The resources that can be used related to personnel are the technicians (provider, protective or victim), the law persons and anyone who will work on the case. On the other hand, actors develop strategy concerning decisions they have to take, based on the training, planning and preparation activities. Planning and organizing the steps an actor will make in case of an incident,

is very productive when the time comes. A well-organized preparation can improve the quality and availability of digital evidence collected and preserved, while minimizing cost and workload [26]. Developing an incident response plan ensures that it was taken under consideration all possible calculated risks [26]. Policies and procedures should be clearly defined and as many likely scenarios should be considered and tested. To support the plans, actors need to have skilled and experienced personnel. Training plays a vital role to all investigations, by minimizing risks and mistakes. CSPs should be responsible to assist and help practitioners and consumers with all the information and evidence found in their infrastructures. They should be willing to provide the right access to potential evidence shortly after a request has been placed, without compromising the privacy and security of their tenants. In other words, CSP is the one who controls all the assets during a forensic investigation. There are three types of assets; hardware, software and data. After collecting the assets using appropriate resources, will lead to the identification of useful evidence. Examining and analyzing the assets with the use of software tools investigators can find evidence to build a case in the court of law. The types of assets that can be transformed to evidence include but are not limited to remote computers, hard discs, deleted files, times and dates associated with modifications, computer names and IP addresses, usernames and passwords, web server logs, windows event logs, registry entries and temporary files, browser history, temporary internet files and cache memory, etc. Assets related to cellular phones could be SIM cards, call logs, contacts, SMS and MMS, calendar, GPS locations and routes. The main objective of documentation concept is to keep the investigation proper documented in order to increase the probabilities of winning a case in a court of law or in an internal investigation. Documentation at the early stages of the incident also helps to keep track of all the actions have been taken and to proceed with different techniques. Any risk analysis or assessment tests performed during the training and preparation should be documented in order to assist the team. All tools, processes, methods and principles performed should be documented properly in order to maintain the chain of custody. Any changes made to the evidence should be also recorded. According to Grispos [28], "a properly maintained chain of custody provides the documentary history for the entire lifetime of evidence discovered during an investigation". To present the evidence in a court as admissible, all the parties (staff, CSPs, third parties) conducted the investigation should record their actions through logs and notes e.g. who handled the evidence, how was it done, did the integrity of the evidence maintained, how was it stored, etc. The last concept identified in the meta-model is the verdict of the jury (the closure). This concept is related to the evidence and in particular with their presentation. When the verdict is announced, the incident either is resolved or an appeal follows. Either way, the strategy should be revised to identify areas of improvement and review methodologies and procedures. Figure 1 summarizes the critical components of the model for assisting cloud forensic process.

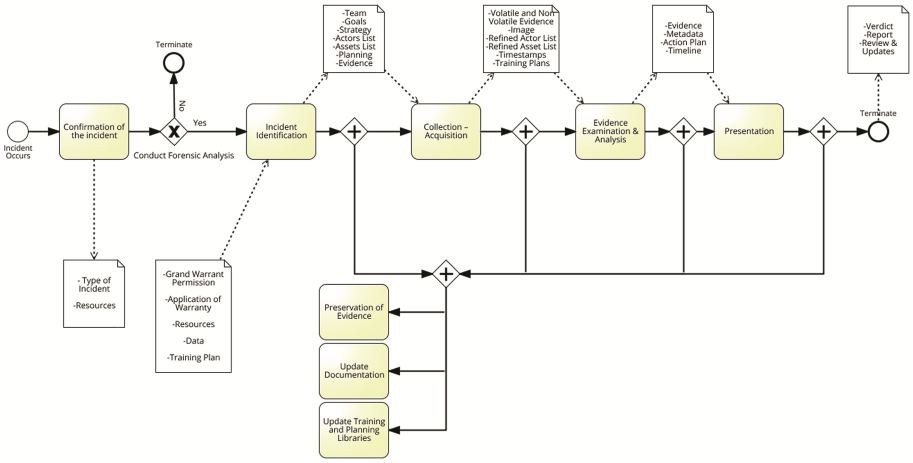


Fig. 1. Process for cloud forensic investigation

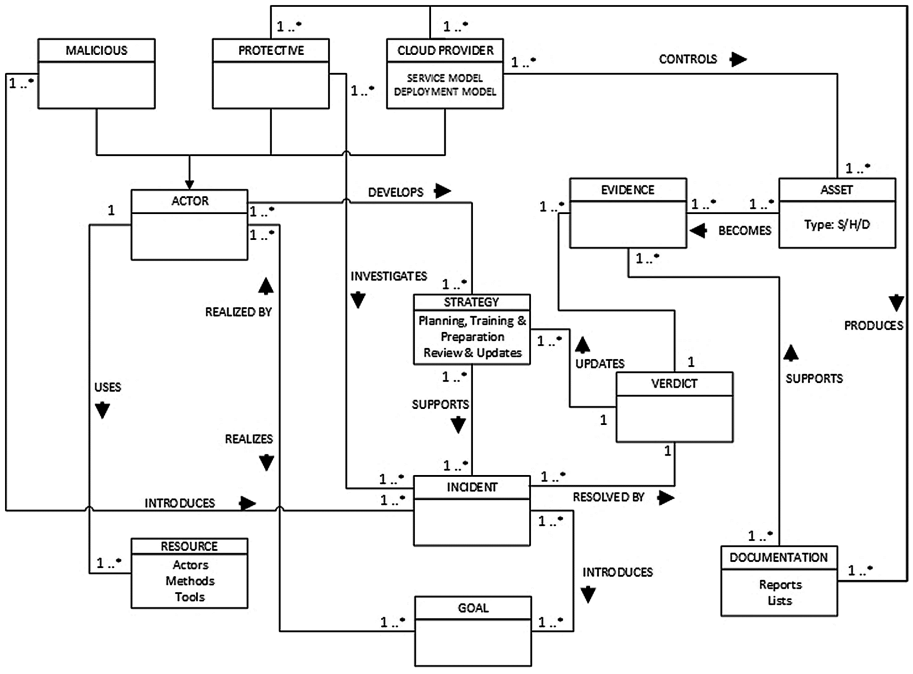


Fig. 2. Meta-model for assisting a cloud forensics process

4 Running Example

For verifying the applicability of the aforementioned meta-model a running example is presented. Through this example a basic analysis is conducted for identifying that all concepts presented are indeed the necessary ones required for describing a specific forensic scenario. The words that match the proposed concepts of the meta-model are marked in bold. The case deals with trafficking illegal digital material in cloud environment.

John, a malicious actor, opens an account with Microsoft Azure Cloud Service Provider (CSP). He registers to use IaaS services. He creates a Virtual Machine (VM) and a webserver where he uploads illegal content of photographs, videos, etc. using the storage (hard disks), Azure is providing. All data is encrypted using cryptographic function and anyone can download the material anonymously as long as is a registered user. Once a day the VM is switched off resulting in the loss of data, leaving it to restart from a clean state. Most of the times John pays the provider with a pay-safe or a prepaid card, thus his ID remains unknown. Protective actors' primary purpose is to find malicious actor and prosecute him.

Incident Confirmation **- John** (Malicious actor) is responsible for the initiation of the **incident** (trafficking illegal content over the internet). The **Cyber Crime Unit** (Protective actors) detects the illegal activity and brings the case into the **head officer** to **decide** whether they are going to proceed into an investigation or not. The head officer is informed about the **type of incident** and the available **resources** and takes the decision to initiate the investigation.

Incident Identification - John uses the cloud, so protective actors locate the **Cloud Provider** that accommodate malicious actor's **servers** and prepare an **application of warranty**. In parallel a special **trained team** responsible for the incident is formed consisting of **IT** and **law officers**. Once the **warrant permission is granted**, a communication with CSP is established and is being asked to **preserve the data**, through **procedures**, which do not suspect the malicious actor. At the same time, protective actors search for previous **similar cases** to identify any common patterns working in parallel on the investigation **strategy** by setting the **goals** and their **initial plan**. The identification of the malicious actor's **IP address** is unsuccessful, due to the third countries proxy servers. Using CSP's assistance, protective actors try to find more evidence such as **card payment information**, cloud providers' **subscriber id's**, **access logs**, **NetFlow records**, **webserver virtual machine** and **cloud storage data** [29]. Any **CSP's personnel** involved in the investigation and their **actions are recorded** and **documented** according to the data preservations procedures and principles. Also, a research is conducted by protective actors to identify the **source of the evidence** and assets, such as **computers, laptops, mobiles**, etc. Once system information and **potential evidence** have been identified with **forensic tools**, protective actors start to implement the **acquisition plan** and produce an **action plan**, **time plan**, **actors'** and **assets' lists**. Due to the fact that the CSP is operating in a different country and the data are stored in data centers geographically spanned in various locations, proper procedures need to be followed to cope with the different jurisdictions. **Trained law officers**, specialized on legal issues, are involved.

Collection–Acquisition - Once the remaining issues relating to jurisdiction have been resolved, the CSP assigns an experienced and skilled technician to produce an **exact copy** of all data of the original media (hard disk) that is under the supervision of the protective actor, using appropriate software such as the **EnCase** or **FTK**. The tools are part of the resources being used to investigate the incident. This operation is followed according to the **training scenarios** that took place during the preparation/training and takes under consideration the acquisition plan. A proper forensic image contains **volatile evidence**, **metadata**, such as, **hashes** and **timestamps** and it compresses all empty blocks. Then, the technician verifies the image for **integrity** and **authenticity** of data by creating MD5 hash values. These tests reveal any alteration of the evidence, in order to use the evidence in a court of law, through forensically acceptable procedures. The problem identified in this process is whether the hired technical staff of the cloud provider has the necessary knowledge and training to properly manage forensic evidence collected from the malicious actor's assets and how trustworthy the whole process is mainly against intentional or accidental data alteration. The **chain of custody** could be considered to be violated with negative results. The entire process of creating the image should be documented in detail, presenting the exact **methods** and tools (resources) that have been used, the produced outputs and the results, a **methodology report**, the technical knowledge of the personnel responsible for the creation, the supervisor's position and any other relevant detail that will help in a lawsuit. With the completion of the controls, the provider sends the image and all data collected to protective actors for examination in order to carry on with the investigation.

Examination–Analysis - Once protective actors receive the VM image and respective data, new checks and controls are taking place to ensure the integrity and validity of the assets. Two **identical copies** are produced to work with and the original one is stored in a secure place with limited access to the head of the investigation. Using appropriate resources (**software tools**), data is being analyzed for any useful information such as files containing **photos**, **videos** and **sounds**, **event logs**, IP addresses, timestamps, etc. At this point, protective actors realize that data is encrypted and a search for finding and identifying **decode keys** is starting. With Azure, where the location of applications and data is abstracted, storing a public key in cloud makes it very difficult to find and retrieve it. File system and **windows registry** is also analyzed. Time is valuable and crucial during an investigation and it is directly related to the amount of data to be analyzed. Let us assume that the CSP managed to produce 20 MB of event logs, 150 MB from NetFlow records, 50 GB of VM snapshot and 1 TB of data. The protective actors load the **VM snapshot** to be able to get more information regarding the structure of the web site and the encryption methods used. The personnel responsible for analyzing the data follows an action plan designed mainly from previous cases. After a thorough investigation protective actors manage to locate and retrieve the decoding keys and the analysis of 1 TB data is starting in order to reveal any evidence. A precise **timeline** with evidence related to the investigation is produced. From the examination of the evidence, protective actors manage to trace malicious actor's IP address. Reports are being produced and handled with all the evidence and techniques followed. The reports contain information about the CSP, the persons involved in the investigation, evidence analysis, methods and techniques followed, respective findings and all

technical terms used. A **final report** is produced by the head of the investigation and presented to the legal authorities.

Presentation - All the stages followed during the above mentioned investigation have been well documented in accordance with forensic principles and procedures, in order to ensure the integrity and the validity of the evidence and to preserve the chain of custody. Before the presentation all evidence, reports, resources used, have been examined thoroughly and tasks have been assigned to experienced personnel who will present the case. Whatever the outcome (**verdict**) of the trial is, all the investigation is **reviewed** from the start and the necessary **updates** have been recorded. Then the case is **closed** and the documentation is stored in a database for future use and training purposes.

5 Conclusions

Undoubtedly, cloud environments are attracting malicious actors to take advantage of the new technology for their criminal activities. Developing cloud services that will assist investigators resolving cyber crime incidents is of vital importance. This work is an initial effort towards this direction. Specifically in this paper a conceptual meta-model was presented in order to deal with the analysis and prospective modeling of the basic concepts a forensic-enabled cloud system. The presentation of a cloud forensic investigation process was crucial for identifying the key aspects of the meta-model in addition to our findings from our previous work on cloud forensics. This paper is an initial step towards the construction of a Cloud Forensic Framework (CFF) that will contain the proposed meta-model as well as a process and the respective design models and tools for assisting designers to reason about the development of forensic enabled cloud systems in order to raise users' trustworthiness on the delivered cloud services.

References

1. RightScale 2015, State of the Cloud Report. http://assets.rightscale.com/uploads/pdfs/RightScale-2015-State-of-the-Cloud-Report.pdf. Accessed Mar 2016
2. IDC Predicts the 3rd Platform. https://www.idc.com/getdoc.jsp?containerId=prUS25285614. Accessed Mar 2016
3. The Global State of Information Security® Survey 2016. http://www.pwc.com/gx/en/issues/cyber-security/information-security-survey.html. Accessed Mar 2016
4. Open Data Center Alliance Cloud Adoption Survey – 2014. http://www.opendatacenteralliance.org/docs/2014MemberSurvey04.pdf. Accessed Mar 2016
5. Simou, S., Kalloniatis, C., Kavakli, E., Gritzalis, S.: Cloud forensics: identifying the major issues and challenges. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 271–284. Springer, Heidelberg (2014)
6. Simou, S., Kalloniatis, C., Mouratidis, H., Gritzalis, S.: Towards the development of a cloud forensics methodology: a conceptual model. In: Persson, A., Stirna, J. (eds.) CAiSE 2015 Workshops. LNBIP, vol. 215, pp. 470–481. Springer, Heidelberg (2015)

7. Simou, S., Kalloniatis, C., Mouratidis, C., Gritzalis, S.: A meta-model for assisting a cloud forensics process. In: Lambrinoudakis, C., Gabillon, A. (eds.) CRiSIS 2015. LNCS, vol. 9572, pp. 177–187. Springer, Heidelberg (2015)
8. Ciardhuáin, S.Ó.: An extended model of cybercrime investigations. Int. J. Digit. Evid. **3**(1), 1–22 (2004)
9. Kohn, M.D., Mariki, M.E., Jan, H.P.E.: Integrated digital forensic process model. Comput. Secur. **38**, 103–115 (2013)
10. ISO/IEC 27037, Information Technology – Security Techniques – Guidelines for Identification, Collection, Acquisition and Preservation of Digital Evidence. http://www.iso.org/iso/catalogue_detail?csnumber=44381. Accessed Mar 2016
11. Zawoad, S., Hasan, R., Skjellum, A.: OCF: an open cloud forensics model for reliable digital forensics. In: 8th International Conference on Cloud Computing (CLOUD), pp. 437–444. IEEE, New York (2015)
12. Simpson, W.R., Chandersekaran, C.: Cloud forensics issues. DTIC document, Institute of Defense Analysis (2014). https://www.ida.org/~/media/Corporate/Files/Publications/IDA_Documents/ITSD/2014/D-5133.ashx. Accessed Mar 2016
13. Cloud Security Alliance: Mapping the Forensic Standard ISO/IEC 27037 to Cloud Computing. CSA Incident Management and Forensics Working Group (2013). https://downloads.cloudsecurityalliance.org/initiatives/imf/Mapping-the-Forensic-Standard-ISO-IEC-27037-to-Cloud-Computing.pdf. Accessed Mar 2016
14. Pichan, A., Lazarescu, M., Soh, S.T.: Cloud forensics: technical challenges, solutions and comparative analysis. Digit. Investig. **13**, 38–57 (2015)
15. Kent, K., Chevalier, S., Grance, T., Dang, H.: Guide to integrating forensic techniques into incident response. NIST Special Publication 800-86 (2006)
16. Martini, B., Choo, K.K.R.: An integrated conceptual digital forensic framework for cloud computing. Digit. Investig. **9**(2), 71–80 (2012)
17. Casey, E., Katz, G., Lewthwaite, J.: Honing digital forensic processes. Digit. Investig. **10**(2), 138–147 (2013)
18. Agarwal, A., Gupta, M., Gupta, S., Gupta, S.C.: Systematic digital forensic investigation model. Int. J. Comput. Sci. Secur. (IJCSS) **5**(1), 118–131 (2011)
19. Williams, J.: ACPO Good Practice Guide for Digital Evidence Version 5.0. Association of Chief Police Officers (2011). http://www.dcs.kcl.ac.uk/staff/richard/7CCSMCFC/ACPO-gpg-digital-evidence-v5.pdf. Accessed Mar 2016
20. von Solms, S., Louwrens, C., Reekie, C., Grobler, T.: A control framework for digital forensics. In: Olivier, M., Shenoi, S. (eds.) Advances in Digital Forensics II, vol. 222, pp. 343–355. Springer, New York (2006)
21. Almulla, S.A., Iraqi, Y., Jones, A.: A state-of-the-art review of cloud forensics. J. Digit. Forensics Secur. Law **9**(4), 22–28 (2014)
22. Vacca, J.R.: Computer Forensics: Computer Crime Scene Investigation. Networking Series. Charles River Media, Inc., Rockland (2005)
23. Braid, M.: Collecting electronic evidence after a system compromise. Australian Computer Emergency Response Team (2001)
24. Prayudi, Y., Sn, A.: Digital chain of custody: state of the art. Int. J. Comput. Appl. **114**(5), 1–9 (2015)
25. Carrier, B., Spafford, E.H.: Getting physical with the digital investigation process. Int. J. Digit. Evid. **2**(2), 1–20 (2003)
26. Beebe, N.L., Clark, J.G.: A hierarchical, objectives-based framework for the digital investigations process. Digit. Investig.: Int. J. Digit. Forensics Incid. Response **2**(2), 147–167 (2005)

27. Aydin, M., Jacob, J.: A comparison of major issues for the development of forensics in cloud computing. In: International Conference on Information Science and Technology (ICIST). IEEE (2013)
28. Grispos, G., Storer, T., Glisson, W.B.: Calm before the storm: the challenges of cloud computing in digital forensics. Int. J. Digit. Crime Forensics (IJDCF) **4**(2), 28–48 (2012). IGI Global, Hershey, PA, USA
29. Dykstra, J., Sherman, A.T.: Understanding issues in cloud forensics: two hypothetical case studies. In: Conference on Digital Forensics, Security and Law, pp. 45–54. Richmond, VA (2011)

Modelling Secure Cloud Computing Systems from a Security Requirements Perspective

Shaun Shei[1(✉)], Christos Kalloniatis[1,2], Haralambos Mouratidis[1], and Aidan Delaney[1]

[1] School of Computing, Engineering and Mathematics, Secure and Dependable Software Systems (SenSe), Research Cluster, University of Brighton, Brighton, UK
{S.Shei,H.Mouratidis,A.J.Delaney}@brighton.ac.uk

[2] Cultural Informatics Laboratory, Department of Cultural Technology and Communication, University of the Aegean, Lesvos, Greece
chkallon@aegean.gr

Abstract. This paper presents a cloud modelling language for defining essential cloud properties, enabling the modelling and reasoning about security issues in cloud environments from a requirements engineering perspective. The relationship between cloud computing and security aspects are described through a meta-model, aligning concepts from cloud computing and security requirements engineering. The central concept of the proposed approach is built around cloud services, where the propagation of relationships from a social perspective, abstract software processes and the foundational infrastructure layer are captured. The proposed concepts are applied on a running example throughout the paper to demonstrate how developers are able to capture and model cloud concepts across multiple conceptual layers, facilitating the understanding of cloud security requirements and the design of security-embedded cloud systems to realise organisational needs.

Keywords: Cloud computing · Cloud security · Cloud security requirements · Modelling language · Security requirements engineering

1 Introduction

Cloud computing enables the provisioning of a wide range of cloud services, delivered on a self-servicing basis for cloud users based on the concept of abstracting physical and virtual computing resources. This paradigm offers seemingly unlimited scalability, availability and flexibility through a pay-per-use model, where users are able to select and deploy cloud services that satisfy their requirements without worrying about how the cloud service is implemented or delivered. However in order to take advantage of these benefits, the distributed nature of the involved technologies implicitly requires the outsourcing of business processes and data to off-premise, third party providers. Thus the users are required to sacrifice a degree of access and control over their data, relying on third party

S. Katsikas et al. (Eds.): TrustBus 2016, LNCS 9830, pp. 48–62, 2016.
DOI: 10.1007/978-3-319-44341-6_4

providers to ensure that their data is kept secure and available. As the concept of cloud computing evolves from utility services to the foundational focus of business IT infrastructure, there is a clear need for ensuring the security of cloud computing systems and maintaining service-provider transparency.

Cloud computing is an evolving term in which the core characteristics has seen numerous reiterations, definitions and is over-saturated in terms of standard definitions [1]. Producing a concrete meaning, reasoning or realisation of cloud computing is hugely dependent on the sector and discipline; between industry, academia, levels of abstraction and granularity, all parties attempts to provide their own definitions [2]. Despite the perceived immaturity of the concept, specifically concerning security, privacy and jurisdictional issues, organisations are still integrating cloud computing as part of their business strategy [3,4]. Some have even acknowledged the numerous issues but have opted to use the technology regardless, citing the need to keep up with competitors and stay relevant in today's industry [5,6].

The primary challenge in cloud computing adoption is the lack of a systematic methodology to facilitate the understanding, reasoning and modelling of non-functional aspects, specifically regarding security issues [7,8]. Combined with a deploy-first, fix-later approach, this creates scenarios resulting in high losses when deploying business systems to the cloud in terms of financial assets, man-hours and reputation [9]. For example moving from a traditional IT environment towards a cloud environment without adequate planning and understanding of the security issues creates systems that are insecure by design, that is the system will inherit both traditional security issues in addition to cloud specific issues. The result is insecure operational systems riddled with vulnerabilities which requires constant patching and even redesigns, where the underlying cause is the lack of a methodological approach for understanding and addressing security issues during the system life-cycle. Thus there is a lack of a holistic modelling language that captures user security requirements and cloud computing properties within a well-defined contextual environment, which satisfies the demand for understanding and realising the requirements for secure cloud-based systems [3,9,10].

In this paper we present a modelling language for defining cloud computing properties, based on capturing and modelling the security requirements of organisational systems and providing case-by-case guidance towards deployment properties in cloud environments. This work is part of an on-going research effort to create a framework for holistically modelling secure cloud computing systems, grounded in security requirements engineering and cloud computing security concepts. The framework consists of the modelling language, a process to systematically apply the concepts to the system-under-design and a tool to facilitate automated security requirement analysis. Our work benefits users involved in the process of securing cloud computing systems, for example organisational stakeholders that wish to migrate aspects of their business system to the cloud, providing guidance for security engineers modelling cloud environments or allowing cloud users to understand the security properties of cloud systems. Therefore in

the running example we introduce the users of our work as organisational stakeholders and cloud security engineers under their employment. Our contributions in this paper are:

- *C1*: Definition of a cloud service to provide abstract and fine-grained description of cloud systems.
- *C2*: Concepts required to holistically model a cloud computing environment through a three layer approach which describes properties at the organisation, application and infrastructure level.
- *C3*: Holistic threat and vulnerability analysis through decomposition and propagation of operationalised security constraints through separate or compositions of conceptual cloud layers.

The rest of the paper is structured as follows. A motivating scenario for migrating hospital processes to the cloud is presented in Sect. 2. The cloud modelling language and cloud computing security concepts are defined in Sect. 3. In Sect. 4 we discuss the respective related work. Finally we conclude the paper in Sect. 5, noting the on-going work and contributions.

2 Health-Care Running Example

The health-care industry is one example where business and organisational goals are enacted through a complex environment, involving multiple facets of technology and stakeholders such as the exchange of data through disparate systems, collaboration between geographically dispersed medical personnel and a rapidly growing repository of electronic records and medical images [11]. Thus there is a need to ensure the availability of medical assets, interoperability between collaborating health-care partners and reducing IT upkeep costs. However, due to the sensitive nature of health-care data, there are rigorous federal regulations such as the Health Insurance Portability and Accountability Act (HIPAA) in the USA and the European Parliament and Council Directive 95/46/EC in the EU surrounding security and privacy in data protection, processing and transit practises. For example in the United States, administrators, doctors and nurses in traditional health-care systems are responsible for ensuring strict HIPAA compliance. However, one of the primary security concerns when moving health-care processes to the cloud is the disseminated responsibility for compliance with key regulations such as HIPAA, due to the unavoidable loss of control over valuable assets and the reliance on third-parties to ensure secure practises are satisfied.

Our running example shown in Fig. 1 describes a scenario where one hospital wishes to partially offload their patient records management system to the cloud, in order to improve the availability and interoperability of the records. This information is captured using organisational goal models with existing security requirements engineering approaches, in this example Secure Tropos [12]. The notation used in this scenario is based around goals represented through rounded rectangles, security constraints represented through octagons, actors represented through circles, resources represented through rectangles and the

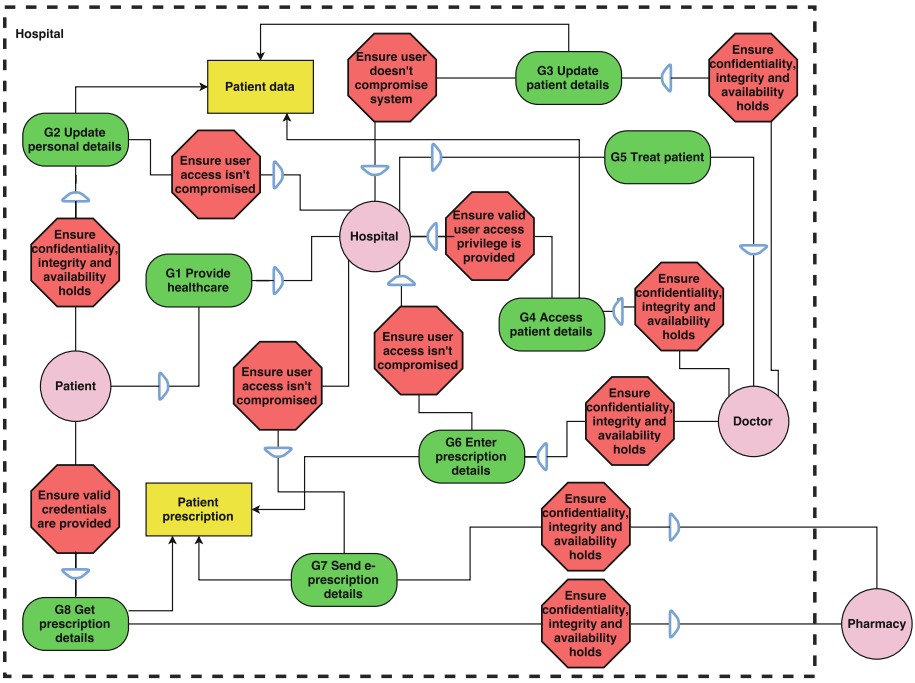


Fig. 1. Simple organisational goal model of hospital processes.

dependency relationship represented through a half circle on the connector. This model indicates the security requirements of actors based on security constraints placed on goals. For example the actor *"Patient"* depends on the actor *"Hospital"* to achieve the goal *"G2 Update personal details"*, where they place the security constraint *"Ensure confidentiality, integrity and availability holds"* on the depender actor *"Hospital"* while the security constraint *"Ensure user access isn't compromised"* is placed on the dependee actor *"Patient"* by the depender actor. However this model is unable to capture cloud computing properties such as cloud services, virtual machines or multi-tenancy.

3 Cloud Modelling Language

In this section we discuss the cloud computing paradigm and how we capture the essential characteristics from a security requirements engineering perspective. The National Institute of Standards and Technology (NIST) provides the following definition for cloud computing: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.", where the cloud model is composed

of "five essential characteristics, three service models, and four deployment models." [13]. We are interested in modelling the service and deployment models, as each service delivery model has a unique set of associated threats and vulnerabilities at the cloud level, while also posing a threat to existing traditional technologies in a cloud environment [14]. The cloud-specific security issues are based on existing work, where our running example demonstrates several cloud threats and vulnerabilities identified by Hashizume et al. [15]. We define our modelling language through established concepts from software security, cloud computing and requirements engineering, combining knowledge from these domains to describe security properties of cloud computing software systems. We follow the Goal-Oriented Requirements Engineering (GORE) approach from the requirements engineering domain [16], where we argue that a cloud service embodies the realisation of a goal. The proposed cloud meta-model is shown in Fig. 2, illustrating the relationships and attributes of concepts required to describe security in cloud computing through a semi-formal UML notation. The meta-model guides the process of modelling cloud computing systems, through the semi-automated instantiation of concepts and attributes with optional user input to broadly capture scenarios with security in mind. The conceptual model is divided in three groups of concepts. The first group (mainly located on the left side of the

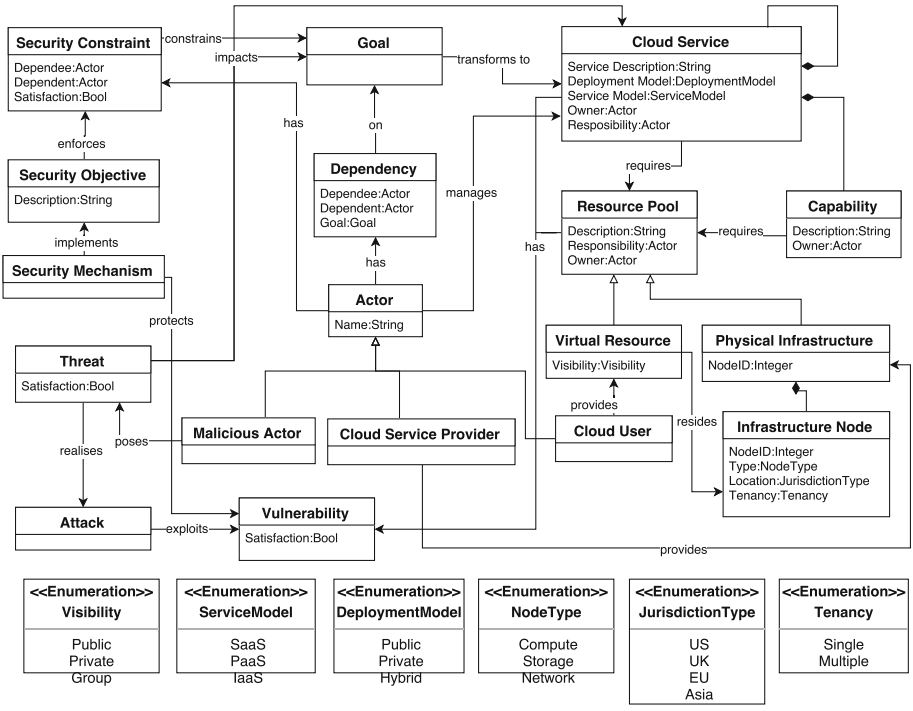


Fig. 2. The cloud meta-model showing the relationships between concepts.

meta-model) represents concepts relating to security requirements engineering. The second group (central part of the meta-model) represents concepts for the requirements engineering analysis whereas the third group (right part of the meta-model) represents the cloud computing concepts. In the following subsections all respective concepts are described based on the aforementioned three groups. For every concept a reference to the running example is provided for better realising the proposed model.

3.1 Security Requirements Engineering Concepts

The proposed concepts are grounded in principles from the requirements engineering and security requirements engineering domain [12,16–18], in order to facilitate the construction of security-ensured software systems from abstract operational needs. The security-oriented aspects of the modelling language are defined below:

Actor: We represent stakeholders, entities or roles using the notion of an actor, drawing from goal-based modelling approaches. The common categories of stakeholders represented as actors are direct or indirect stakeholders, end-users of the system, domain-specialists involved throughout the development process or entities such as components of a system or a physical entity. Referring to the health-care running example, we are able to identify several actors roles within a generic hospital environment. Starting from the list of stakeholders, we identify the notion of patients who wish to receive health-care from the hospital. For example the actor entity *A1: Patient* in the health-care running example represents the patient role. Note that we define the patient as an entity, which refers to the set of stakeholders playing the role of actors with the strategic intention of receiving health-care. Thus our approach defines the notion of actors through roles as opposed to an instantiation of a role, which provides a high-level description of entities in a system and the role they play in relation to other entities. The entity *A2: Hospital* in the running example represents the hospital as a system, for example the actor *A1: Patient* is interested in receiving health-care from the actor *A2: Hospital*. This relationship is represented using the dependency relationship from one actor to another based on a goal, in this case from *A1: Patient* to *A2: Hospital*. Thus the *A1: Patient* depends on the *A2: Hospital* to provide health-care, in which the strategic intent is captured through the notion of an goal. Actors may have one or more strategic interests, which is expressed through the use of the dependency relationship initiated from an actor entity to a goal. The dependency relationship requires an actor as the dependee to initiate the notation, a goal connected from the dependee to the depender and an actor as the depender to complete the relationship.

Malicious Actor: This subset of actor represents a stakeholder with malicious intentions, realised through attacks on the system to exploit vulnerabilities and compromise assets. An example of a malicious actor named *A4: Malicious Actor* is shown in the running example, where they pose the threat *Customer-data manipulation*.

Goal: While there is no uniform notion of a goal in requirements engineering, the general description is a way to achieve different objectives. Thus our notion of a goal is based on representing the strategic interests of stakeholders within the context of a system. We do not explicitly represent goals in our cloud models, instead we define a rule to transform goals into cloud services. Each goal belonging to an organisational model is conceptually mapped as a cloud service entity, thus we transform strategic needs from high-level requirements to a service-oriented entity from a cloud computing perspective. In the health-care running example, the goal *"Receive health-care"* captures the strategic needs of the actor *A1: Patient* which should be achieved under the system. In the health-care example, the goal *"Receive health-care"* has one initiating actor and is connected to one terminal actor through a dependency relationship. This relationship indicates that the actor *A1: Patient* depends on the actor *A2: Hospital* to achieve the goal *"Receive health-care"*, which conceptually represents the responsibility hospitals have for providing health-care to patients.

Threat: A threat embodies the concept of causing harm to an entity, in software security this typically indicates gaining access to, modifying or damaging assets. In the cloud computing context, threats may impact multiple abstract layers. Referring to the running example, the threat *Customer-data manipulation* is posed by a malicious actor, where the threat impacts both *cloud service 1: Patient Details Service* and *cloud service 2: E-prescription Service*. The threat is also realised through the *Cross-site scripting* and *SQL injection* attacks. Threats are unaddressed if at least one vulnerability associated to the threat is not protected by a security mechanism. This is graphically indicated by a exclamation mark inside a red circle, while the satisfaction attribute is flagged false for the instantiated instance of the threat.

Security Constraint: This is a restriction related to security issues, such as the established principles of confidentiality, integrity and availability (CIA). A security constraint is placed from an actor to another actor based around one constrained entity, in this case a goal. This relationship represents the security needs of actors when achieving their goals, which in a cloud computing context indicates security needs that cloud services are required to satisfy. In our running example we have both satisfied and unsatisfied security constraints, respectively indicated visually by a green circle enclosing the letter "s" and a red circle enclosing an exclamation mark. The security constraint *Correct credentials* is satisfied because it is enforced through a security objective, *Ensure only user groups with correct credentials are given access.*

Security Objective: The security objective describes the conditions, criteria and approaches to satisfy security constraints. A high-level description of the security properties provides flexibility when choosing security solutions, as multiple security mechanisms can be implemented to realise the security objective.

Security Mechanism: A security mechanism represents standard security methods for satisfying security objectives, which is described as a high-level solution. In our running example, the security mechanisms *Identity and access*

management and *Dynamic credentials* implements the security objective *Ensure only user groups with correct credentials are given access*. It is the security experts responsibility for selecting and realising security mechanisms during the implementation stage, where they decide the most suitable security mechanism through our cloud models.

Vulnerability: This describes a weakness which allows an attacker to reduce a system's information assurance. An example of a vulnerability is *Insecure interface and APIs*, which impacts both *cloud service 1: Patient Details Service* and *cloud service 2: E-prescription Service*. The vulnerability can be protected by the security mechanism *Web application scanners*, as seen in the running example where the protected vulnerability is visually indicated as satisfied by the letter "s" inside a green circle.

Attack: An attack embodies a specific method of carrying out a threat in order to exploit vulnerabilities in the system. *Cross-site scripting* and *SQL injection* are examples of attacks that exploit the vulnerability *Insecure interface and APIs*, where they are both realised from the threat *Customer-data manipulation*.

3.2 Cloud Computing Concepts

In this section the concepts that are essential for capturing cloud computing properties are presented, centred around the concept of cloud services which are the basic type of resource in a cloud environment.

Cloud Service: A cloud service can be described as a set of six concepts: *capability, actor, resource, relationships, service model, deployment model*. An example of an instantiated cloud service is *cloud service 1: Patient Details Service*, which has the cloud service description *Patient Details Service*, the end-user dependency relationship from the actor *A1: Patient*, the service-provider dependency relationship from the actor *A2: Hospital*, the managed relationship from the actor *A5: CSP*, the requires relationship to virtual resource *Patient Data*, the constraint relationship from the security constraints *SC1: Keep information CIA* and *SC2: Patient access not compromised*, the *SaaS* service model and the *public* deployment model.

Capability: A capability describes, at a high level, an atomic action that is performed by a cloud service to produce a desired outcome. In our running example we have not explicitly modelled capabilities, because the cloud services *cloud service 1: Patient Details Service* and *cloud service 2: E-prescription Service* both provide atomic capabilities. That is if a cloud service only provides an atomic capability, the capability itself is the cloud service and is represented as such. If a cloud service provides two or more capabilities, it is conceptually a composite cloud service where each capability is explicitly represented as an entity belonging to the specific cloud service.

Cloud Actor: A cloud service involves direct and indirect stakeholders, disparately distributed throughout the cloud management levels. We define two specialised roles of actors in the cloud, the cloud user and the cloud service provider.

Resource: Assets are represented through resources, which is essential for understanding cloud computing systems and reasoning about security properties. We define two subtypes of resources: *Virtual Resource* to represent information and intangible data and *Physical Infrastructure* to represent tangible assets. Physical Infrastructure is a conceptual container to hold *Infrastructure Nodes*, which abstractly represents physical computing components such as processing servers, data storage and networking connections.

Cloud Service Model: The cloud service provider provides a high-level description of how a cloud service is delivered, which indicates the level of control and the parties responsible for managing computing components. We include in the model the respective cloud service models; Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) [13]. It is necessary to model the cloud service models due to the impact of model-dependent issues, such as threats which target physical components relevant only to the IaaS model and vulnerabilities which apply to virtual machines in the SaaS model. This also captures the level of control and responsibility of each actor in relation to the interaction with cloud services, which provides transparency and traceability. For example it is possible to indicate that a cloud provider is responsible for managing physical components on a cloud service deployed through a IaaS model, therefore they are responsible for security requirements addressing threats on physical components. The cloud service user has the highest level of control in the IaaS model, a lesser degree of control for the PaaS model and little to no control for the SaaS model. This is indicated through the resources and configurations the user is responsible and has access to, given a cloud service with a designated cloud service model. For example in the IaaS model the cloud provider is responsible for providing and managing the low-level components such as networking, storage, servers and configuring virtualisation to ensure that users of the service are able to manage the high-level components such as the operating system, application and data. Given the same cloud service with a SaaS model, the cloud service user is therefore only responsible for using the service where the cloud provider is responsible for configuring and managing components at the IaaS and PaaS layers in order to deliver the service. e.g. The cloud provider manages and configures virtual machines to ensure that the users of the SaaS receives the same quality of service regardless of resource load or storage limits.

Cloud Deployment Model: The type of deployment models are also necessary for security reasoning. Thus we include the cloud deployment models as public, private, community and hybrid [13]. The deployment model determines the user group, level of access and accessibility of the cloud service. It also explicitly determines the physical location, ownership and management of computing resources such as infrastructure and data.

Relationships: We propose five types of relationships that are required to capture interactions with cloud services:

– **Dependency:** One actor is dependent on another actor to deliver a cloud service. The depender actor is either a cloud user or an end-user. The dependee

actor is a cloud service provider, who themselves can also be a cloud user. A cloud user is an actor that uses an cloud service but has dependents, for example *A2: Hospital* is a cloud service provider and cloud user because they provide the *cloud service 1: Patient Details Service* to the end-user *A1: Patient*, but *A2: Hospital* uses the cloud service provider *A5: CSP* and is dependent on them to provide components of the cloud service.

- **Requires:** One or more resources are required by a cloud service. For example the cloud service *Patient Details Service* requires *Virtual Resource: Patient Data*, indicating that the cloud service requires digital patient records to perform computing processes such as creating, editing and deleting data.
- **Security Constraint:** One or more security constraints are placed on a cloud service. As explained in the security requirements concepts section previously, this represents the security needs of stakeholders which has to be satisfied by the cloud service.
- **Manages:** One or more actors are responsible for managing a cloud service. We use the term manage to represent parties responsible for providing cloud resources, configuring cloud components and ensuring security and jurisdictional requirements are fulfilled. For example *A5: CSP* is responsible for managing the *IaaS*, *PaaS* and *SaaS* layers of the cloud service *Patient Details Service*, while A2: Hospital manages the *SaaS* layer of the same cloud service. This represents that both *A5: CSP* and *A2: Hospital* share the responsibility of ensuring security needs are met and enforced at the *SaaS* level, while only the *A5: CSP* is responsible for the *IaaS* and *PaaS* layers.
- **Impacts:** Threats or vulnerabilities which impact the security properties of a cloud service. These may target a cloud service, or individual capabilities within a cloud service. For example the threat *Customer-data manipulation* and the vulnerability *Insecure interface and APIs* target the cloud services *Patient Details Service* and *E-prescription Service*, which indicates that any entities encapsulated in the cloud services are indirectly impacted.

3.3 Cloud Environment Model

Reasoning about security in the cloud computing environment requires a more detailed procedure due to the high complexity and its multi-parameter nature. For assisting prospective users in modelling secure cloud services we have created visual models of the system-under-design during our process based on the cloud meta-model. The graphical notation of cloud security concepts and relationships help facilitate understanding of complex cloud environments, where cloud security engineers are able to holistically model and evaluate security properties of cloud systems based on three conceptual layers at the organisational, application and infrastructure level. Here we explain the cloud environment model through the running example, instantiating concepts from our meta-model to create a holistic cloud view as shown in Fig. 3. The novelty of the approach is based on the three-layer view which assists designers in capturing the necessary concepts for security reasoning holistically.

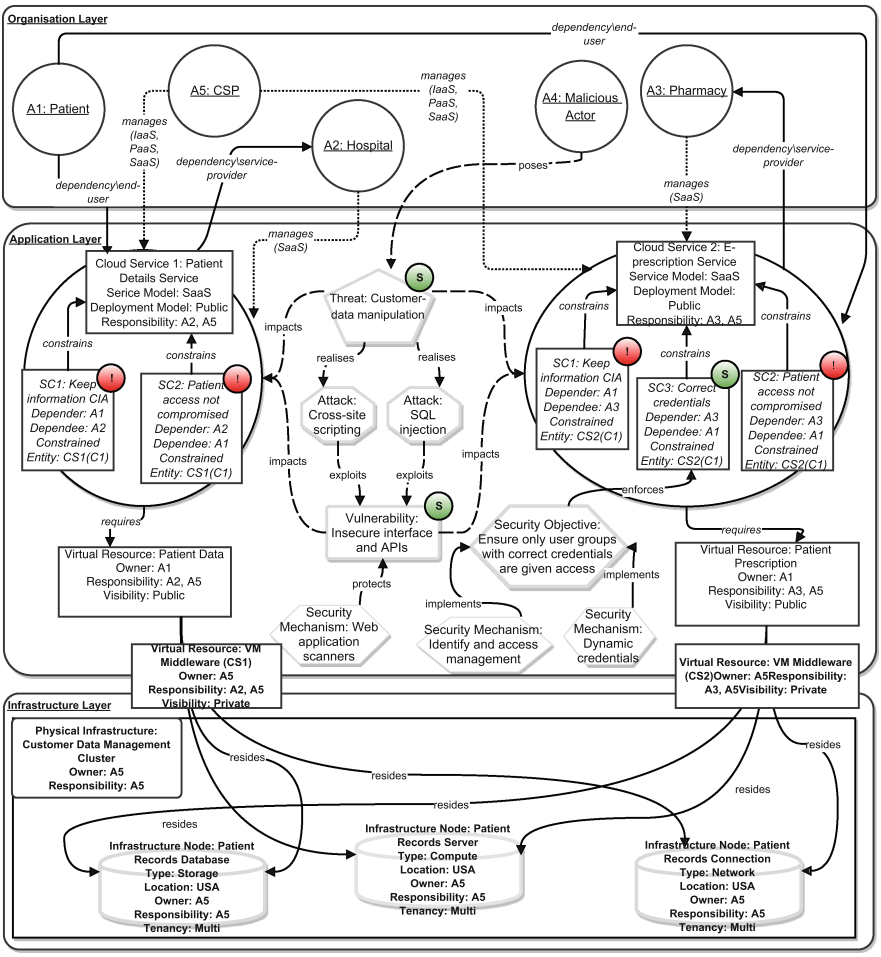


Fig. 3. A holistic cloud view of the health-care running example.

Organisation Concepts: We describe the stakeholders on the organisation layer in the cloud environment model, identifying the direct and indirect stakeholders as actors through their relationship with cloud services. In our running example we have identified five actors; the *A1: Patient* is an end-user of the *cloud service 1: Patient Details Service* and *cloud service 2: E-prescription Service*, *A2: Hospital* manages cloud service 1 and they are a cloud service provider to *A1: Patient* and a cloud user to *A5: CSP*, *A3: Pharmacy* manages the cloud service 2 and is a cloud service provider to *A1: Patient* and a cloud user to *A5: CSP*, *A5: CSP* is a cloud service provider that manages both clouds services at all three service levels and *A4: Malicious Actor* is a malicious actor which poses a security threat *Customer-data manipulation*.

Application Concepts: This layer represents the abstract concepts for software and applications in the system-under-design, centring around cloud services, components interacting with cloud services and the security impacts. In our running example we model two cloud services, the security issues impacting them, the virtual resources they require and partial solutions for mitigation. The service and deployment models of each cloud service determines the actors that owns the cloud service, actors responsible for managing the cloud service, security issues and propagation of dependencies. For example the cloud service *Patient Details Service* uses a SaaS model and is deployed publicly, determining that the CSP actor *A5: CSP* is responsible for managing components on all three service model layers (SaaS, PaaS, IaaS) while the actor *A2: Hospital* manages the SaaS components. *Customer-data manipulation* is a cloud-specific threat impacting all three service model layers [15], therefore the actors responsible for the cloud services impacted by the threat will be held accountable for deploying security mechanisms in order to mitigate identified threats. In this case the *Customer-data manipulation* threat is realised through attacks *Cross-site scripting* and *SQL injection* which exploit the *Insecure interface and APIs* vulnerability, where the cloud security engineer modelling the system has identified a security mechanism *Web application scanners* to protect the vulnerability and thus mitigate the underlying threat.

Infrastructure Concepts: We define this layer to abstractly model physical components required to realise cloud computing services, which we capture as infrastructure nodes belonging to one or more physical infrastructure containers representing IT infrastructure. In our running example, we model a single physical infrastructure to represent one physical IT infrastructure owned and managed by the CSP *A5: CSP*. The compute capabilities are enabled through the abstract notions of a storage, compute and network entity, where they are multi-tenant and geographically located in the USA. From these attributes we can infer jurisdictional legislation such as the USA Patriot Act which applies to all virtual resources residing on infrastructure physically located in the USA, where multi-tenancy indicates that compute processes are physically shared with one or more unknown cloud service users thus also violating HIPAA compliance. In this scenario the cloud security engineer has a range of options for mitigating these issues, one option is to change the service model of the cloud services to IaaS and provision single-tenancy infrastructure nodes from a CSP geographically located outside the US, thus ensuring dedicated access to cloud computing resources in order to comply with HIPAA regulations.

4 Related Work

Existing research in cloud security is primary focused on mitigating mechanisms and software solutions at the implementation level, which targets software systems that are already implemented and operational [19]. While most work covers multiple security sub-areas, they only target these cloud computing issues in isolation, for example considering security properties in software systems or human

factors on a social level but failing to provide direct correlations between the conceptual layers required to fully capture cloud computing issues and indicate impact on security requirements [7,20]. Li et al. provides a holistic security requirements-eliciting approach towards socio-technical systems [21], however this work lacks expressive power for capturing cloud computing-specific properties which is essential for representing cloud security issues, impact and mitigation. Beckers et al. provides a pattern-based approach for eliciting security requirements and selecting security measures in a cloud computing context [22]. While they provide detailed descriptions of cloud components and properties through their Cloud System Analysis Pattern (CSAP), they do not support propagation of threats or directly model the correlation between security issues and how they are addressed through the instantiation of solutions.

The proposed approach ensures that the system-under-design incorporates security from the early requirements stage, thus addressing underlying vulnerabilities and provides a foundation for implementing security mechanisms and enforcing requirements. We achieve this by building upon existing work in security requirements engineering that lacks the capability to capture or reason about cloud-specific security issues from a holistic point of view [12,23]. This is achieved through a systematic approach which describes and examines cloud computing properties from three distinct but essential levels of abstraction, aggregating layer-specific details to generate a holistic view of a cloud environment [9]. We identify cloud-specific threats and the impact of attacks within the context of the cloud computing system to elicit security requirements, which is realised through cloud service configurations.

5 Conclusion

Currently there is a lack of a methodology offering systematic support for the process of realising organisational and business needs security through the cloud computing paradigm. Our work seeks to fill this gap by providing a methodological approach for eliciting secure cloud environment needs from a requirements engineering perspective, enabling developers to realise organisational needs on a cloud computing context with security embedded in the process. For contributions *C1* and *C2* we have defined a language to capture cloud computing concepts that enables the modelling of essential cloud properties required to describe cloud services, which we argue represents both abstractly and through a fine-grained perspective, the organisational needs and the relationships required for achieving them. For *C3* we provide a security-by-design approach using concepts from security requirements engineering, allowing us to model and address cloud security threats and mitigation mechanisms.

We are currently working on a framework to enable the automated transformation of cloud security controls into security patterns, thus providing a pattern library for applying security policies and mechanisms from a security requirements perspective. Initial efforts have been taken to identify patterns from several domains in the Cloud Controls Matrix (CCM) provided by the Cloud Security Alliance (CSA).

References

1. Chen, Y., Paxson, V., Katz, R.H.: . Whats new about cloud computing security. University of California, Berkeley, Report No. UCB/EECS-2010-5, 20 January 2010
2. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput. Commun. Rev. **39**(1), 50–55 (2008)
3. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A.: Cloud computing The business perspective. Decis. Support Syst. **51**(1), 176–189 (2011)
4. Horwath, C., Chan, W., Leung, E., Pili, H.: Enterprise Risk Management for Cloud Computing. COSO, Hoboken (2012)
5. Merrill, T., Kang, T.: Cloud Computing: Is Your Company Weighing Both Benefits & Risks? Ace Group, New York (2014)
6. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud migration research: a systematic review. IEEE Trans. Cloud Comput. **1**(2), 142–157 (2013)
7. Sengupta, S., Kaulgud, V., Sharma, V.S.: Cloud computing security-trends and research directions. In: 2011 IEEE World Congress on Services (SERVICES), pp. 524–531. IEEE, July 2011
8. Takabi, H., Joshi, J.B.D., Ahn, G.J.: Security and privacy challenges in cloud computing environments. IEEE Secur. Priv. **6**, 24–31 (2010)
9. Almorsy, M., Grundy, J., Müller, I.: An analysis of the cloud computing security problem. In: Proceedings of APSEC 2010 Cloud Workshop, Sydney, Australia, 30th November 2010
10. Armbrust, M., Fox, A., Grioffith, R., Joseph, A.D., Katz, R., Konwinski, A., Zaharia, M.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
11. Ahuja, S.P., Mani, S., Zambrano, J.: A survey of the state of cloud computing in healthcare. Netw. Commun. Technol. **1**(2), 12 (2012)
12. Mouratidis, H., Giorgini, P.: Secure tropos: a security-oriented extension of the tropos methodology. Int. J. Softw. Eng. Knowl. Eng. **17**(02), 285–309 (2007)
13. Mell, P., Grance, T.: The NIST definition of cloud computing (2011)
14. Subashini, S., Kavitha, V.: A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl. **34**(1), 1–11 (2011)
15. Hashizume, K., Rosado, D.G., Fernández-Medina, E., Fernandez, E.B.: An analysis of security issues for cloud computing. J. Internet Serv. Appl. **4**(1), 1–13 (2013)
16. Van Lamsweerde, A.: . Goal-oriented requirements engineering: a guided tour. In: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, pp. 249–262. IEEE (2001)
17. Yu, E.: Modelling strategic relationships for process reengineering. Soc. Model. Requir. Eng. **11**, 2011 (2011)
18. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: an agent-oriented software development methodology. Auton. Agents Multi-Agent Syst. **8**(3), 203–236 (2004)
19. Modi, C., Patel, D., Borisaniya, B., Patel, A., Rajarajan, M.: A survey on security issues and solutions at different layers of cloud computing. J. Supercomput. **63**(2), 561–592 (2013)
20. Iankoulova, I., Daneva, M.: . Cloud computing security requirements: a systematic review. In: 2012 Sixth International Conference on Research Challenges in Information Science (RCIS), pp. 1–7. IEEE, May 2012

21. Li, T., Horkoff, J., Beckers, K., Paja, E., Mylopoulos, J.: . A holistic approach to security attack modeling and analysis. In: Proceedings of the Eighth International i* Workshop (2015)
22. Beckers, K., et al.: A structured method for security requirements elicitation concerning the cloud computing domain. Int. J. Secur. Softw. Eng. (IJSSE) **5**(2), 20–43 (2014)
23. Fabian, B., Gürses, S., Heisel, M., Santen, T., Schmidt, H.: A comparison of security requirements engineering methods. Requir. Eng. **15**(1), 7–40 (2010)

Privacy Requirements

Bottom-Up Cell Suppression that Preserves the Missing-at-random Condition

Yoshitaka Kameya(✉) and Kentaro Hayashi

Department of Information Engineering, Meijo University,
1-501 Shiogama-guchi, Tenpaku-ku, Nagoya 468-8502, Japan
ykameya@meijo-u.ac.jp

Abstract. This paper proposes a cell-suppression based k-anonymization method which keeps minimal the loss of utility. The proposed method uses the Kullback-Leibler (KL) divergence as a utility measure derived from the notions developed in the literature of incomplete data analysis, including the missing-at-random (MAR) condition. To be more specific, we plug the KL divergence into an bottom-up, greedy procedure for a local recoding k-anonymization as a cost function which is efficiently computed. We focus on classification datasets and experimental results exhibit that the proposed method yields a small degradation of classification performance when combined with naive Bayes classifiers.

Keywords: k-anonymity · Cell suppression · Missing-at-random Condition

1 Introduction

Generally, in data mining, fine-grained datasets tend to produce sharper, and accordingly, more useful results. However, when the datasets are human-related, such fineness may lead to re-identification of a person and disclosure of his/her privacy. Re-identification is not only possible from explicit identifiers but from a combination of common personal attributes e.g. age and gender. Such attributes are called quasi-identifiers or QIDs. In privacy-preserving data publishing [1,5,22], we often modify QIDs so that both the risk of re-identification and the loss of utility of the dataset are kept minimal at the same time.

k-Anonymity [16,18] is a well-known privacy requirement on a tabular dataset that, for every combination of QIDs existing in a tuple, at least $k-1$ other tuples must have the same combination of QIDs. Under k-anonymity with a sufficiently large k, the risk of re-identification of a person will be small, since its probability is at most $1/k$. Modifying QIDs in the original dataset so that k-anonymity is satisfied is called k-anonymization. k-Anonymity is attractive in its simplicity and intuitiveness, but it is often quite costly in k-anonymization to fully minimize the loss of the utility of the dataset. For instance, minimizing the number of suppressed cells under k-anonymity is NP-hard [14].

Despite such a discouraging formal result, dozens of practical k-anonymization methods have been proposed. One grouping criterion among these methods is the

S. Katsikas et al. (Eds.): TrustBus 2016, LNCS 9830, pp. 65–78, 2016.
DOI: 10.1007/978-3-319-44341-6_5

range to which an anonymization operator is applied. In global recoding [2,13, 16,19], we replace all occurrences of a value with another general value, while in local recoding [7,12,21], we just replace an occurrence of a value independently of other occurrences. Cell suppression is a typical local recoding operator in which we replace a cell value with a null value. One advantage of suppressing cell values over generalizing them is that the former requires no hierarchical knowledge, and another advantage is that there have been statistical tools including classifiers that can work with suppressed (i.e. missing) data.

In this paper, we propose a cell-suppression based k-anonymization method which keeps minimal the loss of utility using the notion from incomplete data analysis, including the missing-at-random (MAR) condition [15,17]. Kifer and Gehrke [11] formulated anonymized datasets in a probabilistic setting and introduced as a utility measure the Kullback-Leibler (KL) divergence between two empirical distributions, one from the original dataset and the other from the anonymized one. One contribution of this paper is to justify their utility measure from the viewpoint of preserving the MAR condition. An underlying key observation here is that anonymization is an artificial, explicit process that forces the original dataset to be ambiguous or incomplete for avoiding re-identification, whereas traditional incomplete data analysis deals with incomplete datasets as they are, assuming a hidden generation process of the datasets [17]. Another contribution is that we plug the KL divergence into an bottom-up, greedy procedure for a local recoding k-anonymization [7,21] as a cost function which is efficiently computed. We focus on classification datasets where different anonymizations are clearly compared from the viewpoint of utility, though the proposed method can also deal with non-classification datasets.

The rest of this paper is outlined as follows. First, we introduce several background notions and notations in Sect. 2. Then, Sect. 3 describes the proposed method in detail. Experimental results are presented in Sect. 4. Section 5 concludes the paper with some discussions on open problems and related work.

2 Background

2.1 Preliminaries

We begin by introducing some background notions and notations used in the paper. The dataset we consider is a tabular classification dataset of size N with M attributes. We also consider a null value $\perp_j$ at the j-th attribute. In addition, $\mathcal{C}$ is a set of pre-defined classes, and $\mathcal{V}_j$ is a set of discrete non-null values of the j-th attribute. Then, a tuple is comprised of M attribute values and a class label from $\mathcal{C}$, i.e. it is an element of $\mathcal{V}'_1 \times \mathcal{V}'_2 \times \cdots \times \mathcal{V}'_M \times \mathcal{C}$, where $\mathcal{V}'_j = \mathcal{V}_j \cup \{\perp_j\}$. A tuple t is written as $(\boldsymbol{y}, c)$, where $\boldsymbol{y}$ of a vector $(y_1, y_2, \ldots, y_M)$ of attribute values. A dataset $\mathcal{D}$ is a multiset $\{t^{(1)}, t^{(2)}, \ldots, t^{(N)}\}$ of tuples. Throughout the paper, i indicates the index of a tuple in a dataset ($1 \leq i \leq N$), and j indicates the index of an attribute ($1 \leq j \leq M$).

Suppressing a non-null attribute value is to replace it with a null value $\perp_j$. It is obvious that suppression is exactly a generalization along a two-level hierarchy

where the top-level corresponds to $\perp_j$, and the bottom-level only includes raw values from $\mathcal{V}_j$. In incomplete data analysis [17], null or suppressed values are called *missing values*. A tuple $t = (\boldsymbol{x}, c)$ is complete if it contains no missing values, i.e. is an element of $\mathcal{V}_1 \times \mathcal{V}_2 \times \cdots \times \mathcal{V}_M \times \mathcal{C}$. A dataset is complete if it has no incomplete tuples. For non-classification datasets, it is sufficient to consider that $\mathcal{C}$ only contains one dummy class.

Given a complete dataset $\mathcal{D}$, we may use statistics such as $N(\boldsymbol{y}, c) = |\{t^{(i)} \in \mathcal{D} \mid t^{(i)} = (\boldsymbol{y}, c)\}|$, $N(c) = |\{t^{(i)} \in \mathcal{D} \mid t^{(i)} = (\cdot, c)\}|$, $N(\boldsymbol{y}) = |\{t^{(i)} \in \mathcal{D} \mid t^{(i)} = (\boldsymbol{y}, \cdot)\}|$, $N(y_j, c) = |\{t^{(i)} \in \mathcal{D} \mid y_j \text{ is the } j\text{-th attribute value of } t^{(i)} = (\cdot, c)\}|$ and so on. In a probabilistic setting, we introduce a probability distribution $p(\boldsymbol{x}, c)$ over complete tuples $(\boldsymbol{x}, c) = (x_1, x_2, \ldots, x_M, c)$ and compute empirical probabilities $\hat{p}(c) = (N(c) + \alpha)/(N + \alpha|\mathcal{C}|)$ and $\hat{p}(x_j \mid c) = (N(x_j, c) + \alpha)/(N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|)$ for each class c and non-null value x_j. Here, α is non-negative number called the pseudo count, and α prevents the empirical probabilities from being zero when $\alpha > 0$. Throughout the paper, we configure $\alpha = 1$, which results in so-called Laplace smoothing. On the other hand, $N(\neg\perp_j, c)$ denotes the sum of the occurrences of non-null values together with class c, i.e. we have $N(\neg\perp_j, c) = \sum_{x \in \mathcal{V}_j, x \neq \perp_j} N(x, c) = N(c) - N(\perp_j, c)$. Furthermore, a null or suppressed value $\perp_j$ means taking any value in $\mathcal{V}_j$, so its (conditional) probability should always be one. Specifically, we have $p(\perp_j \mid c) = \hat{p}(\perp_j \mid c) = 1$.

2.2 *k*-Anonymity

Here, we describe k-anonymity formally with some additional notations. First, for simplicity, we assume that all attributes $\boldsymbol{y}$ except the class label c in a tuple t are QIDs and focus on reducing the risk of re-identification of a person from QIDs. Whereas $\mathcal{D}$ was defined as a *multiset*, it is often convenient to transform $\mathcal{D}$ into a pair of a tuple set $\mathcal{S}$ and a count table $\mathcal{N}$. $\mathcal{S}$ is defined as $\{\boldsymbol{y} \mid (\boldsymbol{y}, c) \in \mathcal{D}\}$, i.e. an *ordinary* set of distinct tuples. The count table $\mathcal{N}$, on the other hand, stores $N(\boldsymbol{y}, c)$, $N(c)$, $N(\boldsymbol{y})$ and $N(y_j, c)$ in the previous section when needed. It is straightforward to generate an equivalent dataset from $\mathcal{S}$ and $\mathcal{N}$. From the settings above, k-anonymity of a dataset $\mathcal{D}$ is restated as $\min_{(\boldsymbol{y}, \cdot) \in \mathcal{S}} N(\boldsymbol{y}) \geq k$.

2.3 Bottom-Up Cell Suppression

In this paper, we adopt a bottom-up, greedy algorithm for local recoding k-anonymization algorithm, which is a simplified adaptation of the one used in [7,21] into the case of cell-suppression in classification datasets. The algorithm, shown in Algorithms 1, 2 and 3, resembles agglomerative clustering.[1]

The ANONYMIZE procedure is the main routine of the algorithm. The procedure takes as input the anonymity threshold k and the original dataset $\mathcal{D}$ and returns a k-anonymized version of $\mathcal{D}$. The tuple set $\mathcal{S}$ and the count table $\mathcal{N}$

[1] Agglomerative clustering is a typical hierachical clustering in which we start with initial clusters containing single tuples and merge the closest pair of clusters in a bottom-up manner [10].

Algorithm 1. ANONYMIZE$(k, \mathcal{D})$

Require: k: the anonymity to achieve, $\mathcal{D}$: the original dataset
1: Construct the tuple set $\mathcal{S}$ and the count table $\mathcal{N}$ from $\mathcal{D}$
2: Obtain the empirical probability function $\hat{p}$ from $\mathcal{S}$ and $\mathcal{N}$
3: **while** $\min_{(\boldsymbol{y},\cdot)\in\mathcal{S}} N(\boldsymbol{y}) < k$ **do**
4: Pick up $t = (\boldsymbol{y}, c)$ such that $N(\boldsymbol{y}) < k$ randomly from $\mathcal{S}$
5: $t^* := \mathrm{argmin}_{t'=(\boldsymbol{y}',c)\in\mathcal{S}} \Gamma(t, t', \hat{p}, \mathcal{N})$
6: $u :=$ SUPPRESS(t, t^*)
7: UPDATE$(u, t, t^*, \mathcal{S}, \mathcal{N})$
8: **end while**
9: Construct $\mathcal{D}'$ from $\mathcal{S}$ and $\mathcal{N}$
10: **return** $\mathcal{D}'$

Algorithm 2. SUPPRESS(t, t')

Require: t, t': tuples of the same class c to be suppressed
1: Let t be $(y_1, y_2, \ldots, y_M, c)$ and t' be $(y'_1, y'_2, \ldots, y'_M, c)$
2: **return** $u = (u_1, u_2, \ldots, u_M, c)$ s.t. $u_j = y_j$ (if $y_j = y'_j$) or $u_j = \perp_j$ (if $y_j \neq y'_j$)

of $\mathcal{D}$ are used inside the procedure (Line 1). Empirical probability function $\hat{p}$ w.r.t. the original dataset $\mathcal{D}$, which will be referred to in computing the suppression cost, is then obtained from $\mathcal{S}$ and $\mathcal{N}$ (Line 2). The procedure repeatedly chooses two tuples and merges them by suppression until no tuple violates the k-anonymity requirement (Lines 3–8). Specifically, we randomly pick up a tuple t from violating tuples (Line 4) and choose the best counterpart t^* of the same class (Line 5) that minimizes the suppression cost Γ in the case of t and t^* being suppressed and merged. The suppression is actually done by the SUPPRESS procedure (Line 6). Then, the UPDATE procedure replaces two chosen tuples (t and t^*) in $\mathcal{S}$ with the merged one (u) and updates the count table $\mathcal{N}$ (Line 7).

The choice of the cell-suppression cost Γ is crucial since it reflects the utility of the dataset which we wish to exploit. One simple cost function is Γ_{ham}, the one based on the Hamming distance, which is computed as:

$$\Gamma_{\mathrm{ham}}(t, t', \hat{p}, \mathcal{N}) \stackrel{\mathrm{def}}{=} N(\boldsymbol{y}, c)H(\boldsymbol{y}, \boldsymbol{u}) + N(\boldsymbol{y}', c)H(\boldsymbol{y}', \boldsymbol{u}), \tag{1}$$

where $t = (\boldsymbol{y}, c)$, $t' = (\boldsymbol{y}', c)$, $u = (\boldsymbol{u}, c)$ is the tuple to be generated by SUPPRESS(t, t'), and $H(\boldsymbol{a}, \boldsymbol{b})$ is the number of conflicting elements between $\boldsymbol{a}$ and $\boldsymbol{b}$ (null values and non-null values are considered distinct). Γ_{ham} is exactly the total number of cells to be suppressed further and does neither use the empirical probabilities $\hat{p}$ in the original dataset nor the current counts from $\mathcal{N}$. We may also use a cost function Γ_{info}, which is based on information loss [7]:[2]

$$\Gamma_{\mathrm{info}}(t, t', \hat{p}, \mathcal{N}) \stackrel{\mathrm{def}}{=} -\sum_{j:y_j\neq y'_j} \left(N(\boldsymbol{y}, c) \log \hat{p}(y_j \mid c) + N(\boldsymbol{y}', c) \log \hat{p}(y'_j \mid c)\right), \tag{2}$$

[2] To be precise, the original definition by Harada et al. [7] does not consider classification datasets.

Algorithm 3. UPDATE$(u, t, t', \mathcal{S}, \mathcal{N})$

Require: u: a new tuple, t and t': old tuples, $\mathcal{S}$: a tuple set, $\mathcal{N}$: a count table
1: Remove t and t from $\mathcal{S}$
2: Let u be $(\boldsymbol{u}, c)$, t be $(\boldsymbol{y}, c)$ and t' be $(\boldsymbol{y}', c)$
3: **if** $u \in \mathcal{S}$ **then**
4: $\quad N(\boldsymbol{u}, c) := N(\boldsymbol{u}, c) + N(\boldsymbol{y}, c) + N(\boldsymbol{y}', c)$
5: **else**
6: $\quad N(\boldsymbol{u}, c) := N(\boldsymbol{y}, c) + N(\boldsymbol{y}', c)$
7: $\quad \mathcal{S} := \mathcal{S} \cup \{u\}$
8: **end if**
9: Remove all entries of $\mathcal{N}$ concerning t and t'

where $t = (y_1, y_2, \ldots, y_M, c)$ and $t' = (y'_1, y'_2, \ldots, y'_M, c)$. Γ_{info} uses empirical probabilities $\hat{p}(y_j \mid c)$ (y_j is a non-null value x_j or a null value $\perp_j$) computed from the original dataset as shown in Sect. 2.1. The term $-\log \hat{p}(y_j \mid c)$ is the self-information of the j-th attribute taking y_j. Since the self-information of the j-th attribute taking $\perp_j$ is $-\log \hat{p}(\perp_j \mid c) = -\log 1 = 0$, replacing a non-null value x_j with $\perp_j$ loses the information $-\log \hat{p}(x_j \mid c)$. As a result, $\Gamma_{\text{info}}(t, t', \hat{p}, \mathcal{N})$ measures the total amount of information loss in suppressing and merging t and t'. Obviously, the k-anonymization procedure in Sect. 2.3 tends to suppress frequent attribute values when combined with Γ_{info}.

3 The Proposed Method

As said before, we propose a cell-suppression based k-anonymization method which keeps minimal the loss of utility using the notion from incomplete data analysis. In this method, we consider that anonymization is an artificial process that forces the original dataset $\mathcal{D}$ to be ambiguous so as to avoid re-identification of persons. It is then desirable to control such an anonymization process for ensuring the soundness of later statistical inferences such as classification. In the literature of incomplete data analysis, it is proved that, under the missing-at-random (MAR) condition [15,17], the process where some portion of the original dataset $\mathcal{D}$ turns to be missing is ignorable in the inference related to the empirical probability distribution of $\mathcal{D}$. In our context, the MAR condition allows us to obtain empirical probabilities from an anonymized dataset ignoring the anonymization process without distortion.

From the observations above, our k-anonymization method attempts to preserve the MAR condition as well as possible. More precisely, we present a cell-suppression cost function reflecting the deviation from the MAR condition and use it in the k-anonymization procedure introduced in Sect. 2.3. To measure the deviation from the MAR condition, we consider the Kullback-Leibler (KL) divergence in naive Bayes classifiers. In the rest of this section, we will describe these relevant notions in turn.

3.1 Naive Bayes Classification

In classification, we use Naive Bayes [20] as a primary classifier. Naive Bayes assumes that attributes in a classification dataset are conditionally independent of each other, given the class. Despite its strong independence assumption, naive Bayes often works surprisingly well in classifying real datasets [4]. Formally, it is assumed that the probability that a complete tuple $t = (\boldsymbol{x}, c) = (x_1, x_2, \ldots, x_M, c)$ occurs is simplified as $p(t) = p(\boldsymbol{x}, c) = p(c) \prod_j p(x_j \mid c)$. Typically, classification is performed in two steps: we first learn the empirical probabilities $\hat{p}(c)$ and $\hat{p}(x_j \mid c)$ from the complete training dataset, and then predict the most plausible class $c^* = \mathrm{argmax}_{c \in \mathcal{C}}\ \hat{p}(c \mid \boldsymbol{x}) = \mathrm{argmax}_c\ \hat{p}(c) \prod_j\ \hat{p}(x_j \mid c)$ for an unseen data having attribute values $\boldsymbol{x}$.

The independence assumption in naive Bayes also makes it simple to handle incomplete data. That is, noting that $p(\bot_j \mid c) = 1$, the probability that an incomplete tuple $(y_1, y_2, \ldots, y_M, c)$ occurs, where y_j is a non-null value from $\mathcal{V}_j$ or a null value $\bot_j$, is obtained as $p(c) \prod_{j: y_j \neq \bot_j} p(y_j \mid c)$, where null values are all ignored. Similarly, one may learn the empirical probabilities $\hat{p}(x_j \mid c)$ as described in Sect. 2.1 for a non-null value x_j, as if there are no missing values from the beginning. This is a standard way of learning called maximum likelihood (ML) estimation,[3] which is also applicable to anonymized datasets. However, in general, justifying ML estimation requires some extra condition on the process how missing data are generated. The MAR condition explained next is one of such conditions.

3.2 The Missing-at-random Condition

The Process of Anonymization. As said before, under the MAR condition, a standard learning procedure of naive Bayes classifiers is justified even with anonymized datasets. Conversely, to obtain a naive Bayes without distortion brought by anonymization, it is reasonable to anonymize the original dataset so that the MAR condition is preserved.

First, let us model our anonymization process by an analogy to the process of generating missing data [17]. We focus on classification datasets where no class labels will be missing or suppressed. Given an original dataset $\mathcal{D}$ having a complete tuple $(\boldsymbol{x}, c) = (x_1, x_2, \ldots, x_M, c)$, we may anonymize it into an incomplete dataset having $(\boldsymbol{y}, c) = (y_1, y_2, \ldots, y_M, c)$ by suppressing some part of $\boldsymbol{x}$. A binary indicator $\boldsymbol{r} = (r_1, r_2, \ldots, r_M)$ says which part has been suppressed, i.e. $y_j = x_j$ iff $r_j = 1$, or $y_j = \bot_j$ iff $r_j = 0$. Note that, given an incomplete attribute values $\boldsymbol{y}$, the indicator $\boldsymbol{r}$ is uniquely determined. The joint probability of the whole anonymization process behind $\boldsymbol{y}$ is then introduced and we decompose it into two factors:[4]

$$p(\boldsymbol{r}, \boldsymbol{x}, c \mid \theta, \phi) = p(\boldsymbol{r} \mid \boldsymbol{x}, c, \phi) p(\boldsymbol{x}, c \mid \theta). \tag{3}$$

[3] To be precise, learning empirical probabilities using the pseudo count α, shown in Sect. 2.1, is called maximum a posteriori (MAP) estimation. ML estimation is a special case of MAP estimation where $\alpha = 0$. The following discussions can be easily extended to the case of MAP estimation.

[4] Joint distributions decomposed in this way are called selection models [17].

Here, $p(\boldsymbol{x}, c \mid \theta)$ is the probability that a complete, original tuple $(\boldsymbol{x}, c)$ occurs, and $p(\boldsymbol{r} \mid \boldsymbol{x}, c, \phi)$ is the probability that the suppressed pattern is $\boldsymbol{r}$ given such a complete tuple. The latter is called *the missing-data mechanism* and models our choice in anonymization. θ denotes the parameters of the probability distribution over complete tuples, and ϕ denotes the parameters for the missing-data mechanism. Since anonymization is an artificial operation subsequently performed after the original dataset has been obtained, it is natural to think that there is no overlap between θ and ϕ.

Learning Under the MAR Condition. Given an incomplete values $\boldsymbol{y}$, we define $\boldsymbol{x}_{\text{obs}}$ as a collection of x_j's where $r_j = 1$, and $\boldsymbol{x}_{\text{mis}}$ as a collection of x_j's where $r_j = 0$. Thus, $\boldsymbol{x}_{\text{obs}}$ (resp. $\boldsymbol{x}_{\text{mis}}$) denotes the observed or non-suppressed (resp. missing or suppressed) part of $\boldsymbol{x}$. The probability that an incomplete tuple $(\boldsymbol{y}, c)$ occurs is then computed as $p(\boldsymbol{y}, c \mid \theta, \phi) = \sum_{\boldsymbol{x}_{\text{mis}}} p(\boldsymbol{r}, \boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, c \mid \theta, \phi)$ where $\boldsymbol{r}$ is compatible with $\boldsymbol{y}$, and $\boldsymbol{x} = (\boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}})$.

Next, let us consider the procedure for learning a naive Bayes classifier from the dataset $\{(\boldsymbol{y}, c)\}$ which contains only one tuple.[5] As said earlier, one standard learning procedure is ML estimation, where we attempt to maximize the likelihood of the whole process $L(\theta, \phi) = p(\boldsymbol{y}, c)$ by adjusting the parameters θ and ϕ. In other words, we obtain $(\hat{\theta}, \hat{\phi}) = \text{argmax}_{\theta,\phi} L(\theta, \phi)$. Now we assume that the MAR condition is satisfied. The MAR condition states that *the choice in suppression does not depend on the value to be suppressed itself.* This condition is formally written as $\forall \boldsymbol{x}, c\, p(\boldsymbol{r} \mid \boldsymbol{x}, c, \phi) = p(\boldsymbol{r} \mid \boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, c, \phi) = p(\boldsymbol{r} \mid \boldsymbol{x}_{\text{obs}}, c, \phi)$. Then, the likelihood $L(\phi, \theta)$ is transformed as follows:

$$
\begin{aligned}
L(\phi, \theta) &= p(\boldsymbol{y}, c \mid \phi, \theta) = \textstyle\sum_{\boldsymbol{x}_{\text{mis}}} p(\boldsymbol{r}, \boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, c \mid \phi, \theta) \\
&= \textstyle\sum_{\boldsymbol{x}_{\text{mis}}} p(\boldsymbol{r} \mid \boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, c, \phi) p(\boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, c \mid \theta) && (4) \\
&= \textstyle\sum_{\boldsymbol{x}_{\text{mis}}} p(\boldsymbol{r} \mid \boldsymbol{x}_{\text{obs}}, c, \phi) p(\boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, c \mid \theta) && (5) \\
&= p(\boldsymbol{r} \mid \boldsymbol{x}_{\text{obs}}, c, \phi) \textstyle\sum_{\boldsymbol{x}_{\text{mis}}} p(\boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, c \mid \theta) \\
&= p(\boldsymbol{r} \mid \boldsymbol{x}_{\text{obs}}, c, \phi) L'(\theta), && (6)
\end{aligned}
$$

where $L'(\theta) = \sum_{\boldsymbol{x}_{\text{mis}}} p(\boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, c \mid \theta) = p(\boldsymbol{x}_{\text{obs}}, c \mid \theta)$ is the likelihood of the anonymized dataset, ignoring the anonymization process. The MAR condition derives Eq. 5 from Eq. 4. Since $p(\boldsymbol{r} \mid \boldsymbol{x}_{\text{obs}}, \phi)$ is constant w.r.t. θ, Eq. 6 says that, for any ϕ, maximizing $L(\phi, \theta)$ and maximizing $L'(\theta)$ yield the same parameters $\hat{\theta}$, i.e. our anonymization is not influential on learning $\hat{\theta}$, as long as the MAR condition is preserved. The probability $p(\ldots \mid \hat{\theta})$ under the learned parameters $\hat{\theta}$ coincides with the empirical probability $\hat{p}(\ldots)$ used throughout the paper.

[5] Extending the discussion to the case with multiple i.i.d. (independent and identically distributed) tuples $\{(\boldsymbol{y}^{(1)}, c^{(1)}), (\boldsymbol{y}^{(2)}, c^{(2)}), \ldots, (\boldsymbol{y}^{(N)}, c^{(N)})\}$ is fairly straightforward, since the likelihood can be transformed as $L(\theta, \phi) = \prod_i p(\boldsymbol{y}^{(i)}, c^{(i)}) = (\prod_i p(\boldsymbol{r}^{(i)} \mid \boldsymbol{x}^{(i)}, c^{(i)}, \phi))(\prod_i p(\boldsymbol{x}^{(i)}, c^{(i)} \mid \theta))$, where $\boldsymbol{x}^{(i)}$ is the original of $\boldsymbol{y}^{(i)}$.

The KL Divergence for Examining the MAR Condition. The next question is how to preserve the MAR condition in anonymization. First, the MAR condition $\forall \boldsymbol{x}, c\ p(\boldsymbol{r} \mid \boldsymbol{x}_{\mathrm{obs}}, \boldsymbol{x}_{\mathrm{mis}}, c, \phi) = p(\boldsymbol{r} \mid \boldsymbol{x}_{\mathrm{obs}}, c, \phi)$ can always be rewritten as $\forall \boldsymbol{x}, c\ p(\boldsymbol{x}_{\mathrm{mis}} \mid \boldsymbol{x}_{\mathrm{obs}}, \boldsymbol{r}, c, \phi) = p(\boldsymbol{x}_{\mathrm{mis}} \mid \boldsymbol{x}_{\mathrm{obs}}, c, \phi)$. By the independence assumption in naive Bayes, this is simplified as $\forall \boldsymbol{x}_{\mathrm{mis}}, c\, p(\boldsymbol{x}_{\mathrm{mis}} \mid \boldsymbol{r}_{\mathrm{mis}} = \boldsymbol{0}, c, \phi) = p(\boldsymbol{x}_{\mathrm{mis}} \mid c, \phi)$, where $\boldsymbol{x}_{\mathrm{obs}}$ and $\boldsymbol{x}_{\mathrm{mis}}$ are independent given c, and $\boldsymbol{r}_{\mathrm{mis}}$ is the portion of $\boldsymbol{r}$ corresponding to $\boldsymbol{x}_{\mathrm{mis}}$ which is necessarily all zero. Furthermore, this statement is satisfied when $p(x_j \mid r_j = 0, c, \phi) = p(x_j \mid c, \phi)$ for all x_j such that $r_j = 0$ (y_j is a suppressed value), using naive Bayes's assumption again. The resulting statement says that the missing part of the j-th attribute must follow the same distribution as the one over all j-th attribute values of original tuples. Since the observed part and the missing part are mutually exclusive and collectively exhaustive, this statement must also apply to the observed part. Eventually we see that, when the empirical distribution from the original dataset is identical to those from an anonymized dataset, the MAR condition is preserved.

To measure the deviation from the MAR condition, we consider the Kullback-Leibler (KL) divergence, which was firstly introduced by Kifer and Gehrke [11] in the literature of anonymization. The KL divergence is defined and simplified under the independence assumption in naive Bayes:

$$\mathrm{KL}(\hat{p}, \hat{q}) = \sum_{\boldsymbol{x}, c} \hat{p}(\boldsymbol{x}, c) \log \frac{\hat{p}(\boldsymbol{x}, c)}{\hat{q}(\boldsymbol{x}, c)} = \sum_{c} \hat{p}(c) \sum_{j} \sum_{x_j} \hat{p}(x_j \mid c) \log \frac{\hat{p}(x_j \mid c)}{\hat{q}(x_j \mid c)} \tag{7}$$

$$= \sum_{c} \hat{p}(c) \sum_{j} \mathrm{KL}_{j,c}(\hat{p}, \hat{q}) \ \text{ where } \ \mathrm{KL}_{j,c}(\hat{p}, \hat{q}) = \sum_{x_j} \hat{p}(x_j \mid c) \log \frac{\hat{p}(x_j \mid c)}{\hat{q}(x_j \mid c)}$$

(the derivation is presented in the appendix). Here $\hat{p}$ is the empirical probability distribution from the original dataset, and $\hat{q}$ is the one from an anonymized dataset, which may be unfinished one in the Anonymize procedure (Sect. 2.3). $\mathrm{KL}_{j,c}(\hat{p}, \hat{q})$ is the class- and attribute-wise version of the KL divergence. It is known that the KL divergence is non-negative, and hence making $\mathrm{KL}(\hat{p}, \hat{q})$ smaller implies making each $\mathrm{KL}_{j,c}(\hat{p}, \hat{q})$ smaller. This further implies making $\hat{p}(x_j \mid c)$ and $\hat{q}(x_j \mid c)$ closer, which leads to the preservation of the MAR condition. In addition, the summation in Eq. 7 is taken over all classes and distinct attribute values and so is costly to compute. Next, we plug the KL divergence above into the Anonymize procedure as a new cost function which is efficiently computed.

3.3 Cell-Suppression Cost for Preserving the MAR Condition

To introduce a light-weight cost function that reflects the MAR condition, we consider the difference between two KL divergences before and after a cell suppression in the Anonymize procedure. Cell suppression is a local operator, so the difference between these two quantities will also be rather limited.

More formally, let $\mathcal{D}^{(\ell)}$ be the dataset obtained at the end of the ℓ-th loop in the Anonymize procedure. We apply a cell suppression once to the dataset at

each loop. The difference is then written as $\Delta\mathrm{KL} = \mathrm{KL}(\hat{p}, \hat{q}') - \mathrm{KL}(\hat{p}, \hat{q})$, where $\hat{p}$ is the empirical distribution from the original dataset $\mathcal{D} = \mathcal{D}^{(0)}$, $\hat{q}$ is the one from $\mathcal{D}^{(\ell)}$, and $\hat{q}'$ is the one from $\mathcal{D}^{(\ell+1)}$ ($\ell \geq 0$). Here we easily have:

$$\Delta\mathrm{KL} = \sum_c \hat{p}(c) \sum_j \Delta\mathrm{KL}_{j,c} \text{ where } \Delta\mathrm{KL}_{j,c} = \sum_{x_j} \hat{p}(x_j \mid c) \log \frac{\hat{q}(x_j \mid c)}{\hat{q}'(x_j \mid c)}. \quad (8)$$

Let us consider next a more specific case in which the j-th non-null attribute value x_j of a tuple $t = (\boldsymbol{y}, c)$ is suppressed in $\mathcal{D}^{(\ell)}$. Also suppose that $\hat{q}(x_j \mid c)$ has been obtained from $\mathcal{D}^{(\ell)}$ as $(N(x_j, c)+\alpha)/(N(\neg\perp_j, c)+\alpha|\mathcal{V}_j|)$. Then, $\hat{q}'(x_j \mid c)$ is obtained from $\mathcal{D}^{(\ell+1)}$ as $(N(x_j, c) - N(\boldsymbol{y}, c) + \alpha)/(N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|)$, in which the count of the suppressed non-null value is decreased by $N(\boldsymbol{y}, c)$. Substituting these empirical probabilities into Eq. 8 results in:

$$\Delta\mathrm{KL}_{j,c} = \hat{p}(x_j \mid c) \log \frac{N(x_j, c) + \alpha}{N(x_j, c) - N(\boldsymbol{y}, c) + \alpha} + \log \frac{N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|}{N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|} \quad (9)$$

(the derivation is presented in the appendix).

Based on the above, consider an extended case where two incomplete tuples $t = (y_1, y_2, \ldots, y_M, c)$ and $t' = (y'_1, y'_2, \ldots, y'_M, c)$ are suppressed and merged. Suppressions occur at y_j and/or y'_j in the j-th attribute such that y_j and y'_j are distinct. An extension of Eq. 9 to this case is derived as:

$$\begin{aligned}\Delta\mathrm{KL}_{j,c} = \hat{p}(y_j \mid c) \log \frac{N(y_j, c) + \alpha}{N(y_j, c) - w_j(t) + \alpha} + \hat{p}(y'_j \mid c) \log \frac{N(y'_j, c) + \alpha}{N(y'_j, c) - w_j(t') + \alpha} \\ + \log \frac{N(\neg\perp_j, c) - (w_j(t) + w_j(t')) + \alpha|\mathcal{V}_j|}{N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|}. \quad (10)\end{aligned}$$

Note here that $\hat{p}(\perp_j \mid c) = 1$ and we define $w_j(t) = N(\boldsymbol{y}, c)$ (if $y_j \neq \perp_j$) or $w_j(t) = 0$ (if $y_j = \perp_j$) for an incomplete tuple $t = (\boldsymbol{y}, c)$. Finally, a cost function Γ_{mar} which measures the deviation from the MAR condition is introduced as:

$$\Gamma_{\mathrm{mar}}(t, t', \hat{p}, \mathcal{N}) \stackrel{\mathrm{def}}{=} \Delta\mathrm{KL} = \textstyle\sum_c \hat{p}(c) \sum_j \Delta\mathrm{KL}_{j,c}, \quad (11)$$

where $\Delta\mathrm{KL}_{j,c}$ is the one defined in Eq. 10. One may find from Eq. 10 that we have only to refer to the quantities related to the suppressed attribute values and hence computing $\Delta\mathrm{KL}_{j,c}$ just requires a constant time. Lastly, we see from Eqs. 8 and 11 that Γ_{mar} can be negative. This case happens when the empirical distribution $\hat{q}'$ after the suppression gets closer than $\hat{q}$ to the original one $\hat{p}$.

4 Experimental Results

We tested the proposed method using the adult dataset available from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Adult). Specifically, we compare cost functions Γ_{ham}, Γ_{info} and Γ_{mar} plugged into the

ANONYMIZE procedure in Sect. 2.3. We additionally introduced a cost function Γ_{hybrid} as a simple hybrid of Γ_{ham} and Γ_{mar}, defined as follows:

$$\Gamma_{\mathrm{hybrid}}(t,t',\hat{p},\mathcal{N}) \stackrel{\mathrm{def}}{=} \begin{cases} \Gamma_{\mathrm{mar}}(t,t',\hat{p},\mathcal{N})/\Gamma_{\mathrm{ham}}(t,t',\hat{p},\mathcal{N}) & (\Gamma_{\mathrm{mar}}(t,t',\hat{p},\mathcal{N}) \leq 0) \\ \Gamma_{\mathrm{mar}}(t,t',\hat{p},\mathcal{N})\Gamma_{\mathrm{ham}}(t,t',\hat{p},\mathcal{N}) & (\Gamma_{\mathrm{mar}}(t,t',\hat{p},\mathcal{N}) > 0). \end{cases} \tag{12}$$

In this hybrid function, Γ_{mar} works as a base cost function, and Γ_{ham} plays a role of a penalty function which increases the cost according to the Hamming distance, i.e. the total number of suppressed cells.

The adult dataset has two classes: salary above or below 50,000 dollars. Furthermore, following the previous work [2,9,19], we used eight attributes for a person: *age*, *work class*, *education*, *marital status*, *occupation*, *race*, *gender* and *native country*. All attribute except *age* are discrete, and we discretized the *age* attribute as $[15,20)$, $[20,25)$, $[25,30)$, $\ldots$, $[70,75)$, $[75,80)$ and $[80,95)$, where we first split the whole range into the ranges of five years and then merged the last three to ensure that each range includes more than 100 tuples.

The classifiers we used are naive Bayes classifiers and C4.5, implemented in Weka [20]. Each classifier is evaluated by average error rate in stratified 10-fold cross validation. Before evaluation, we first anonymized the original datasets, and in each fold of cross validation, we use the anonymized version for the training dataset and the original version for the test dataset. All classifiers were run under Weka's default setting. Since the ANONYMIZE procedure runs in a randomized way, the obtained results were averaged over 30 trials.

Figure 1 (left) shows the KL divergence between the empirical distribution from the original dataset and the one from the datasets k-anonymized by the ANONYMIZE procedure with $k = 2, 5, 10, 15, \ldots$ and cost functions. Figure 1 (right) shows the number of suppressed cells in the k-anonymized datasets. In all graphs presented in the paper, the lines labeled ham, info, mar and hybrid correspond to the cases with Γ_{ham}, Γ_{info}, Γ_{mar} and Γ_{hybrid}, respectively. It is

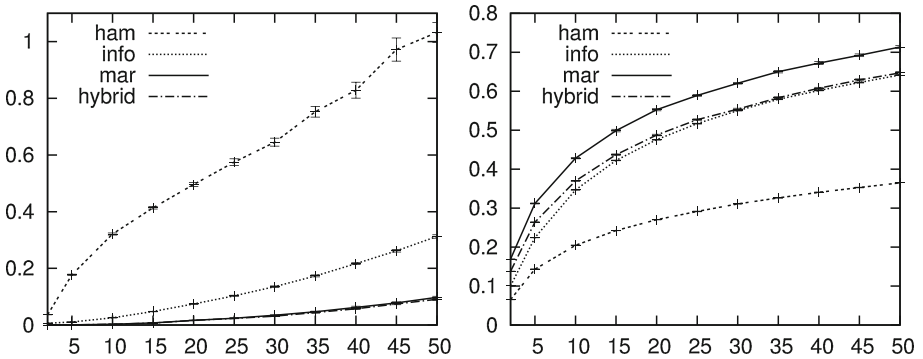


Fig. 1. The KL divergence (left) and the ratio of suppressed cells (right) in k-anonymized datasets, with various k (indicated by the x-axis) and cost functions. In the case of the KL divergence, lines mar and hybrid overlap almost entirely.

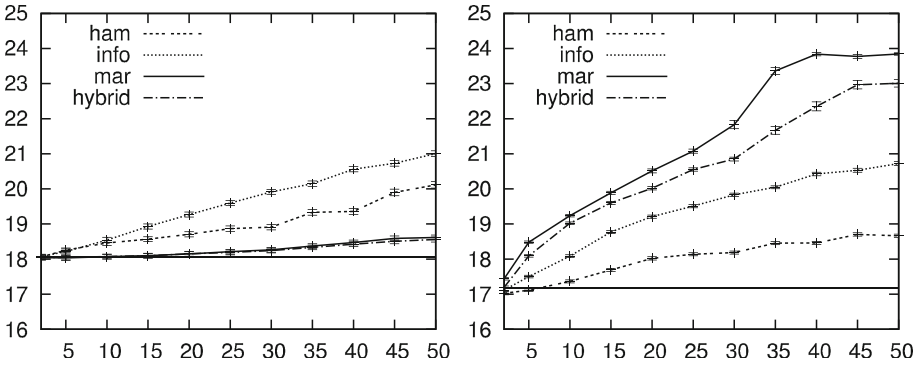


Fig. 2. The average error rate (%) of naive Bayes (left) and C4.5 (right) for k-anonymized datasets, with various k (the x-axis) and cost functions. In the case of naive Bayes, lines mar and hybrid overlap almost entirely.

found in Fig. 1 (left) that, as expected, the KL divergence is smaller with Γ_{mar} and Γ_{hybrid}. Figure 1 (right), on the other hand, exhibits a contrasting behavior that Γ_{ham} yields a smaller number of suppressed cells, which is also expected. In addition, the error bars in the graphs indicate the 95 % confidence intervals. The error bars are narrow, so we can see that the ANONYMIZE procedure works stably.

Figure 2 shows the average error rate (%) of naive Bayes (left) and C4.5 (right) for k-anonymized datasets. The horizontal line indicates the average error rate for the original dataset. In these graphs, Γ_{mar} and Γ_{hybrid} give the least degradation of error rate when combined with naive Bayes, as the theory suggests. However, C4.5 did not work well with Γ_{mar}. From the fact that Γ_{hybrid} which brings less suppressions reduces error rate, the number of suppressed cells seems to give a highly negative impact on the classification performance of C4.5.

5 Concluding Remarks

This paper proposed a cell-suppression based k-anonymization method which keeps minimal the loss of utility. The proposed method aims to preserve the MAR condition and uses the KL divergence as a utility measure. From the discussions and the experimental results presented in this paper, our approach is shown to be statistically promising in both formal and practical senses. On the other hand, there remain a couple of open problems. Here we conclude the paper by discussing such open problems and related work in the literature.

First, a newly introduced cost function Γ_{mar}, which is based on the KL divergence and the independence assumption in naive Bayes, only considers attributes individually. This also applies to most of the existing work, e.g. Γ_{info} used in [7], but some authors take multi-dimensional approaches, in which two or more attributes are jointly taken into account. For instance, given a classification dataset,

kACTUS [12] performs cell suppression based on a decision tree built in advance. Relaxing the independence assumption in naive Bayes would be one possible extension of the proposed method.

There have been several methods targeting classification datasets. Many of such methods [2,6,9,19], as well as kACTUS above, exploit classification-centric heuristic scores such as information gain. Although out target is not limited to classification datasets, as a simple hybrid cost Γ_{hybrid} used in our experiment suggests, some classification-centric cost function would contribute to the improvement of classification performance. In addition, anonymization may be performed in big data environments consisting of, for example, data providers, data collectors and data users who have different requirements [22]. To balance several cost functions, multi-objective optimization techniques look attractive. Dewri et al. [3] explored an evolutionary multi-objective optimization to determine a suitable anonymity threshold k.

As mentioned earlier, one advantage of cell suppression is that it requires no hierarchical knowledge. If such knowledge is available, the coarsening-at-random (CAR) condition [8] would be a key notion since it is a generalization of the MAR condition considering partial information loss in each cell. In addition, Harada et al. proposed a way for automatically constructing hierarchical knowledge [7].

Appendix: Derivation of the Proposed Suppression Cost

Here, we complete the derivation of the cost function Γ_{mar} by showing how to obtain Eqs. 7 and 9. First, let us note that $\hat{p}(c) = \hat{q}(c)$ holds since the class label c is initially non-null and will be never suppressed. Equation 7 is then derived as follows:

$$
\begin{aligned}
&\mathrm{KL}(\hat{p}, \hat{q}) \\
&= \sum_{\boldsymbol{x},c} \hat{p}(\boldsymbol{x}, c) \log \frac{\hat{p}(\boldsymbol{x}, c)}{\hat{q}(\boldsymbol{x}, c)} = \sum_{\boldsymbol{x},c} \Big(\hat{p}(c) \prod_{j'=1}^{M} \hat{p}(x_{j'} \mid c) \Big) \log \frac{\hat{p}(c) \prod_{j=1}^{M} \hat{p}(x_j \mid c)}{\hat{q}(c) \prod_{j=1}^{M} \hat{q}(x_j \mid c)} \\
&= \sum_{c} \hat{p}(c) \sum_{x_1} \sum_{x_2} \cdots \sum_{x_M} \Big(\prod_{j'=1}^{M} \hat{p}(x_{j'} \mid c) \Big) \sum_{j=1}^{M} \log \frac{\hat{p}(x_j \mid c)}{\hat{q}(x_j \mid c)} \\
&= \sum_{c} \hat{p}(c) \sum_{j=1}^{M} \sum_{x_1} \cdots \sum_{x_{j-1}} \sum_{x_j} \sum_{x_{j+1}} \cdots \sum_{x_M} \\
&\qquad \Big(\prod_{j'=1}^{j-1} \hat{p}(x_{j'} \mid c) \Big) \hat{p}(x_j \mid c) \Big(\prod_{j'=j+1}^{M} \hat{p}(x_{j'} \mid c) \Big) \log \frac{\hat{p}(x_j \mid c)}{\hat{q}(x_j \mid c)} \qquad (13) \\
&= \sum_{c} \hat{p}(c) \sum_{j=1}^{M} \sum_{x_j} \hat{p}(x_j \mid c) \log \frac{\hat{p}(x_j \mid c)}{\hat{q}(x_j \mid c)} \cdot \\
&\qquad \sum_{x_1} \cdots \sum_{x_{j-1}} \sum_{x_{j+1}} \cdots \sum_{x_M} \Big(\prod_{j'=1}^{j-1} \hat{p}(x_{j'} \mid c) \Big) \Big(\prod_{j'=j+1}^{M} \hat{p}(x_{j'} \mid c) \Big) \qquad (14)
\end{aligned}
$$

$$= \sum_c \hat{p}(c) \sum_{j=1}^{M} \sum_{x_j} \hat{p}(x_j \mid c) \log \frac{\hat{p}(x_j \mid c)}{\hat{q}(x_j \mid c)} \Big(\prod_{j'=1}^{j-1} \sum_{x_{j'}} \hat{p}(x_{j'} \mid c) \Big) \Big(\prod_{j'=j+1}^{M} \sum_{x_{j'}} \hat{p}(x_{j'} \mid c) \Big)$$

$$= \sum_c \hat{p}(c) \sum_{j=1}^{M} \sum_{x_j} \hat{p}(x_j \mid c) \log \frac{\hat{p}(x_j \mid c)}{\hat{q}(x_j \mid c)}. \tag{15}$$

In Eqs. 13 and 14, we carefully reordered summations and moved irrelevant factors outside the summations wherever possible. Equation 15 was finally derived using $\sum_{x_{j'}} \hat{p}(x_{j'} \mid c) = 1$ since $\hat{p}$ is a probability function.

On the other hand, for Eq. 9, we have been considering a specific case where the j-th non-null attribute value x_j of a tuple $t = (\boldsymbol{y}, c)$ is suppressed. We have $\hat{q}(x_j \mid c) = (N(x_j, c) + \alpha)/(N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|)$ and $\hat{q}'(x_j \mid c) = (N(x_j, c) - N(\boldsymbol{y}, c) + \alpha)/(N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|)$ as already mentioned, and additionally, for each value x'_j of j-th attribute which is not suppressed this time (i.e. $x'_j \neq x_j$), we have $\hat{q}'(x'_j \mid c) = (N(x'_j, c) + \alpha)/(N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|)$. Substituting these probabilities into Eq. 8 results in Eq. 9 as follows:

$$\begin{aligned}
&\Delta\mathrm{KL}_{j,c} \\
&= \hat{p}(x_j \mid c) \log \frac{\hat{q}(x_j \mid c)}{\hat{q}'(x_j \mid c)} + \sum_{x'_j : x'_j \neq x_j} \hat{p}(x'_j \mid c) \log \frac{\hat{q}(x'_j \mid c)}{\hat{q}'(x'_j \mid c)} \\
&= \hat{p}(x_j \mid c) \log \left(\frac{N(x_j, c) + \alpha}{N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|} \cdot \frac{N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|}{N(x_j, c) - N(\boldsymbol{y}, c) + \alpha} \right) + \\
&\qquad \sum_{x'_j : x'_j \neq x_j} \hat{p}(x'_j \mid c) \log \left(\frac{N(x'_j, c) + \alpha}{N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|} \cdot \frac{N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|}{N(x'_j, c) + \alpha} \right) \\
&= \hat{p}(x_j \mid c) \log \frac{N(x_j, c) + \alpha}{N(x_j, c) - N(\boldsymbol{y}, c) + \alpha} + \hat{p}(x_j \mid c) \log \frac{N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|}{N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|} + \\
&\qquad \sum_{x'_j : x'_j \neq x_j} \hat{p}(x'_j \mid c) \log \frac{N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|}{N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|} \\
&= \hat{p}(x_j \mid c) \log \frac{N(x_j, c) + \alpha}{N(x_j, c) - N(\boldsymbol{y}, c) + \alpha} + \\
&\qquad \log \frac{N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|}{N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|} \left(\hat{p}(x_j \mid c) + \sum_{x'_j : x'_j \neq x_j} \hat{p}(x'_j \mid c) \right) \\
&= \hat{p}(x_j \mid c) \log \frac{N(x_j, c) + \alpha}{N(x_j, c) - N(\boldsymbol{y}, c) + \alpha} + \log \frac{N(\neg\perp_j, c) - N(\boldsymbol{y}, c) + \alpha|\mathcal{V}_j|}{N(\neg\perp_j, c) + \alpha|\mathcal{V}_j|}.
\end{aligned}$$

References

1. Aggarwal, C.C.: Data Mining: The Textbook. Springer, Switzerland (2015)
2. Bayardo, R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proceedings of ICDE-05, pp. 217–228 (2005)
3. Dewri, R., Ray, I., Ray, I., Whitley, D.: On the optimal selection of k in the k-anonymity problem. In: Proceedings of ICDE-08, pp. 1364–1366 (2008)

4. Domingos, P., Pazzani, M.: On the optimality of the simple Bayesian classifier under zero-one loss. Mach. Learn. **29**, 103–130 (1997)
5. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: a survey of recent developments. ACM Comput. Surv. **42**(4), 14:1–14:53 (2010)
6. Fung, B.C.M., Wang, K., Yu, P.S.: Anonymizing classification data for privacy preservatio. IEEE Trans. Knowl. Data Eng. **19**(5), 711–725 (2007)
7. Harada, K., Sato, Y., Togashi, Y.: Reducing amount of information loss in k-anonymization for secondary use of collected personal information. In: Proceedings of the 2012 Service Research and Innovation Institute Global Conference, pp. 61–69 (2012)
8. Heitjan, D.F.: Ignorability and coarse data. Ann. Stat. **19**(4), 2244–2253 (1991)
9. Iyengar, V.: Transforming data to satisfy privacy constraints. In: Proceedings of KDD-02, pp. 279–288 (2002)
10. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
11. Kifer, D., Gehrke, J.: Injecting utility into anonymized datasets. In: Proceedings of SIGMOD-06, pp. 217–228 (2006)
12. Kisilevich, S., Rokach, L., Elovici, Y.: Efficient multidimensional suppresion for k-anonymity. IEEE Trans. Knowl. Data Eng. **22**(3), 334–347 (2010)
13. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: efficient full-domain k-anonymity. In: Proceedings of SIGMOD-05, pp. 49–60 (2005)
14. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: Proceedings of PODS-04, pp. 223–228 (2004)
15. Rubin, D.B.: Inference and missing data. Biometrika **63**, 581–592 (1976)
16. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. **13**(6), 670–682 (2001)
17. Schafer, J.L., Graham, J.W.: Missing data: our view of the state of the art. Psychol. Methods **7**, 147–177 (2002)
18. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. Int. J. Uncertainty Fuzziness Knowl. Based Syst. **10**(5), 571–588 (2002)
19. Wang, K., Yu, P.S., Chakraborty, S.: Bottom-up generalization: a data mining solution to privacy protection. In: Proceedings of ICDM-04, pp. 249–256 (2004)
20. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
21. Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., Fu, A.W.C.: Utility-based anonymization using local recoding. In: Proceedings of KDD-06, pp. 785–790 (2006)
22. Xu, L., Jiang, C., Chen, Y., Wang, J., Ren, Y.: A framework for categorizing and applying privacy-preservation techniques in big data mining. Computer **49**(2), 54–62 (2016)

Understanding the Privacy Goal Intervenability

Rene Meis(✉) and Maritta Heisel

paluno - The Ruhr Institute for Software Technology,
University of Duisburg-Essen, Duisburg, Germany
{rene.meis,maritta.heisel}@paluno.uni-due.de

Abstract. Privacy is gaining more and more attention in society and hence, gains more importance as a software quality that has to be considered during software development. A privacy goal that has not yet been deeply studied is the empowerment of end-users to have control over how their personal data is processed by information systems. This privacy goal is called intervenability. Several surveys have shown that one of end-users' main privacy concerns is the lack of intervenability options in information systems. In this paper, we refine the privacy goal intervenability into a software requirements taxonomy and relate it to a taxonomy of transparency requirements because transparency can be regarded as a prerequisite for intervenability. The combined taxonomy of intervenability and transparency requirements shall guide requirements engineers to identify the intervenability requirements relevant for the system they consider. We validated the completeness of our taxonomy by comparing it to the relevant literature that we derived based on a systematic literature review.

1 Introduction

A central concern of end-users with regard to privacy is that they have almost no control over their personal data once these are put into an information system [1–4]. End-users wish for more empowerment, i.e. they want to keep the control over their personal data and how their data is processed by information systems. Hansen [5] summarizes this and other privacy needs into the privacy goal *intervenability*. Hansen states *"Intervenability aims at the possibility for parties involved in any privacy-relevant data processing to interfere with the ongoing or planned data processing. The objective of intervenability is the application of corrective measures and counterbalances where necessary."* [5].

Intervenability is a complex software quality that is strongly coupled with other privacy-related goals. For example, end-users have to be sufficiently aware of how and what personal data is processed and which options exist to intervene in order to be able to exercise these options. Hence, the privacy goal transparency can be seen as prerequisite for intervenability.

This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG) under grant No. GRK 2167, Research Training Group "User-Centered Social Media".

S. Katsikas et al. (Eds.): TrustBus 2016, LNCS 9830, pp. 79–94, 2016.
DOI: 10.1007/978-3-319-44341-6_6

As a first step to assist requirements engineers to deal with the complex privacy goal intervenability, we propose a requirements taxonomy that further refines intervenability into subrequirements enriched with attributes and associated to transparency requirements that we identified in [6]. The taxonomy shall help requirements engineers to understand which intervenability and transparency requirements have to be considered for the system they analyze.

The rest of the paper is structured as follows. Our privacy requirements taxonomy is derived and presented in Sect. 2 and validated using related work identified using a systematic literature review in Sect. 3. Section 4 concludes the paper.

2 Deriving and Structuring Requirements on Intervenability

In Sect. 2.1, we systematically analyze the privacy principles described by the international standard ISO/IEC 29100:2011 [7] and the draft of the EU data protection regulation [8] to derive the intervenability requirements they contain and the transparency requirements related to them. To derive the requirements, we analyze the description of the privacy principles and the formulations of the regulation. We keep the formulation of the identified intervenability and transparency requirements close to the original documents from which we identified them. In Sect. 2.1, we enumerate these derived requirements using the notation I*n* for intervenability requirements and T*n* for the related transparency requirements. As the ISO principles and EU articles partly overlap, we identified several refinements of identified requirements. We relate those requirements using a *refines* relation. If an intervenability requirement In_1 refines a part of another requirement In_2, this means that In_1 adds further details on how or which possibilities have to exist to intervene in the processing of personal data. Furthermore, we identified that there are transparency requirements that are closely related to intervenability requirements. This is, because in order to be able to make use of intervenability mechanisms, data subjects have to be aware of them. Hence, we use a *relatedTo* relation to make the relations between transparency and intervenability requirements explicit. The *refines* (directed dashed edges) and *relatedTo* (solid edges) relation are visualized as an initial overview of intervenability requirements in Fig. 1. In Sect. 2.2, we structure the intervenability requirements identified in Sect. 2.1 into a taxonomy of intervenability requirements and integrate this taxonomy into the taxonomy of transparency requirements introduced in [6]. The taxonomy is presented as an extensible metamodel using a UML class diagram.

ISO/IEC 29100:2011 and the draft of the EU data protection regulation do not use the same terminology. To avoid ambiguities, we use the following term definitions from the draft of the EU data protection regulation in this paper.

Data subject *"means an identified natural person or a natural person who can be identified, directly or indirectly, by means reasonably likely to be used by*

the controller or by any other natural or legal person, [...].” This term is called *PII principal* in ISO/IEC 29100:2011.

Personal data *“means any information relating to a data subject.”* This term is called *personally identifiable information (PII)* in ISO/IEC 29100:2011.

Processing *“means any operation or set of operations which is performed upon personal data or sets of personal data, whether or not by automated means, such as collection, recording, organization, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, erasure or destruction.”*

Controller *“means the natural or legal person, public authority, agency or any other body which alone or jointly with others determines the purposes, conditions and means of the processing of personal data; [...].”* This term is called *PII controller* in ISO/IEC 29100:2011.

Supervisory authority *“means a public authority which is established by a Member State in accordance with Article 46.”* Article 46 states that supervisory authorities *“are responsible for monitoring the application of this Regulation and for contributing to its consistent application throughout the Union, [...].”*

2.1 Requirements Identification from Privacy Principles and Legislation

ISO/IEC 29100 Privacy Principles. To derive our taxonomy of intervenability requirements, we first consider the international standard ISO/IEC 29100:2011 [7], which defines 11 privacy principles which are a superset of the OECD principles [9] and the US fair information practices (FIPs) [10].

We start our analysis with the *consent and choice principle*, which is obviously concerned with providing data subjects the power to decide how their data is processed. From this principle, we obtain the following intervenability and transparency requirements.

I1 Present to the data subjects the choice whether or not to allow the processing of their personal data.

I2 Obtain the opt-in consent of the data subject for collecting or otherwise processing sensitive personal data.

T1 Inform data subjects before obtaining consent about their rights to access their personal data and to influence the processing of these.

I3 Provide data subjects with the opportunity to choose how their personal data is handled.

I4 Allow data subjects to withdraw consent easily and free of charge.

T2 Where the personal data processing is not based on consent but instead on another legal basis, the data subject should be notified wherever possible.

I5 Where the data subject has the ability to withdraw consent and has chosen to do so, these personal data should be exempted from processing for any purpose not legally mandated.

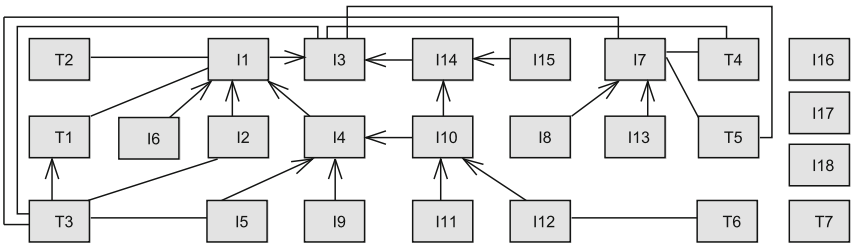


Fig. 1. Initial overview of intervenability requirements

I6 Provide data subjects with clear, prominent, easily understandable, accessible and affordable mechanisms to exercise choice and to give consent in relation to the processing of their personal data at the time of collection, first use or as soon as practicable thereafter.

Requirement I3 states that data subjects shall have the opportunity to choose how their data is handled and is the most general intervenability requirement. It is refined by I1 (cf. Fig. 1) that states that data subjects shall have the choice whether their data is processed or not. I1 is further refined by I2 that requires opt-in consent for processing of sensitive personal data, I4 that requires the possibility to withdraw consent, and I6 that describes requirements for the mechanisms to realize I1. I5 refines I4 by describing the effects of withdrawing consent. Both transparency requirements T1 and T2 are related to I1 (cf. Fig. 1). T1 requires that data subjects have to be informed about their rights before consent is obtained. T2 requires to inform data subjects if their data is processed without their explicit consent.

From the *openness, transparency and notice principle* we identify an additional transparency requirement that is related to all intervenability requirements that describe the choices and means for data subjects to influence how their data is processed (cf. Fig. 1).

T3 Disclose the choices and means offered by the controller to data subjects for the purposes of limiting the processing of, and for accessing, correcting and removing their information.

The following two intervenability requirements are derived from the *individual participation and access principle*.

I7 Give data subjects the ability to access and review their personal data, provided their identity is first authenticated with an appropriate level of assurance and such access is not prohibited by applicable law.

I8 Allow data subjects to challenge the accuracy and completeness of their personal data and have it amended, corrected or removed as appropriate and possible in the specific context.

I7 and I8 are not refinements of the already identified intervenability requirements, because they are not concerned with how data subjects can influence

how or if their personal data is processed. But we consider I8 as a kind of refinement of I7, because I8 depends on I7. Note that I7 prescribes that data subjects shall be empowered with the ability to access and review their personal data. Hence, I7 is considered as an intervenability requirement. Nevertheless, allowing data subjects to access and review their personal data also contributes to transparency.

The other principles presented in ISO 29100 do not contain further statements from which we can derive intervenability requirements.

Draft of the EU Data Protection Regulation. To identify further intervenability and transparency requirements and to refine the already identified requirements, we analyze the draft of the EU data protection regulation[1] [8]. We selected this regulation as a representative data protection regulation. In contrast to the situation in the US where no privacy regulations covering all industrial branches exist, the EU data protection regulation covers all industrial branches.

Article 7 describes the conditions for consent and we derive from it the following intervenability requirement that refines I4.

I9 The data subject shall have the right to withdraw his or her consent at any time.

Article 12 specifies requirements on mechanisms for exercising the rights of data subjects. We identified the following two transparency requirements that are related to all intervenability requirements that describe the choices and means for data subjects to influence how their data is processed.

T4 The controller shall inform the data subject without delay and, at the latest within one month of receipt of the request, whether or not any action has been taken if a data subject requested information and shall provide the requested information.

T5 If the controller refuses to take action on the request of the data subject, the controller shall inform the data subject of the reasons for the refusal and on the possibilities of lodging a complaint to the supervisory authority and seeking a judicial remedy.

Article 17 is about the right to be forgotten and to erasure. From this article we derive the following requirements.

I10 The data subject shall have the right to obtain from the controller the erasure of personal data relating to them and the abstention from further dissemination of such data if the data subject withdraws consent or objects to the processing of personal data.

[1] The draft of the EU data protection regulation was adopted with some changes on 27 April 2016 and entered into force on 24 May 2016. Note that our analysis is based on the draft and not on the final version of the regulation.

I11 The controller shall carry out the erasure without delay, except to the extent that the retention of the personal data is necessary.
I12 Where erasure is not possible, the controller shall instead restrict processing of personal data.
T6 The controller shall inform the data subject before lifting the restriction on processing.

I10, I11, and I12 refine the consequence of withdrawing consent (I4) and objecting to processing (I14 see below). T6 requires that data subjects are informed about the restrictions on processing implied by I12 before these are lifted.

The right to data portability is introduced by Article 18. It implies the following intervenability requirement that refines I7.

I13 The data subject shall have the right, where personal data are processed by electronic means and in a structured and commonly used format, to obtain from the controller a copy of data undergoing processing in an electronic and structured format which is commonly used and allows for further use by the data subject.

Article 19 describes the right to object. From this we derived the following two intervenability requirements that refine I3.

I14 The data subject shall have the right to object, on grounds relating to their particular situation, at any time to the processing of personal data, unless the controller demonstrates compelling legitimate grounds for the processing.
I15 If the objection is valid, the controller shall no longer use or otherwise process the personal data concerned.

Article 53 describes the powers of supervisory authorities. In contrast to the previously identified requirements, the following requirements do not describe intervention possibilities for data subjects or needs to provide information to data subjects, but for/to supervisory authorities.

T7 Supervisory authorities may order the controller to provide any information relevant for the performance of their duties to them.
I16 Supervisory authorities may order the rectification or erasure of all data when they have been processed in breach of the provisions of a regulation.
I17 Supervisory authorities may impose a temporary or definitive ban on processing.
I18 Supervisory authorities may order to suspend data flows to a recipient in a third country or to an international organization.

Table 1 summarizes from which ISO 29100 principles and articles of the draft of the EU data protection regulation which initial intervenability and transparency requirements were derived. Additionally, it allows to associate the elements of our intervenability requirements taxonomy (introduced in the next section) with the principles and articles from which these were identified.

Table 1. Mapping of ISO principles and data protection articles to the requirements

Principle/Article	I*n*/T*n*	IR	DIR	AIR	PIR	EIR	IIR
Consent and choice	I1–I6, T1, T2	X	X		X		
Openness, transparency and notice	T3		X		X		
Individual participation and access	I7, I8	X	X				
Article 7	I9		X				
Article 12	T4, T5						X
Article 17	I10–I12, T6	X	X				X
Article 18	I13	X	X				
Article 19	I14, I15	X	X				
Article 53	I16–I18, T7	X		X		X	

IR: IntervenabilityRequirement DIR: DataSubjectInterventionRequirement
AIR: AuthorityInterventionRequirement PIR: ProcessingInformationRequirement
EIR: ExceptionalInformationRequirement IIR: InterventionInformationRequirement

2.2 Setting up an Intervenability Requirements Taxonomy

We now structure the identified preliminary intervenability requirements into an intervenability requirements taxonomy. We integrate this taxonomy into the transparency requirements taxonomy presented in earlier work [6] using the related preliminary transparency requirements. Figure 2 shows our taxonomy in the form of a metamodel using a UML class diagram. Note that we only show the attributes and enumerations of the transparency taxonomy that are relevant for this paper. All elements that have bold font and thick lines are newly added to the transparency taxonomy. The requirements with dark gray background represent the newly identified transparency and intervenability requirements.

Table 2 provides an overview of how the initial requirements are reflected in our proposed taxonomy. In the following, we explain the new elements of our taxonomy and how they are related to the requirements introduced in [6].

Intervenability Requirement. The root element of our intervenability requirements taxonomy is the *IntervenabilityRequirement*. We modeled it as an abstract class because only its specializations shall be instantiated. It contains the attribute effect that describes the consequences of an intervenability requirement. The possible effects are derived from the preliminary requirements I1, I3, I5, I7, I8, I10–I13, and I15–I18, and are summarized in the enumeration InterventionEffect (cf. Fig. 2). The effects are that data subjects get *access* to their personal data, that their personal data is *not processed*, that the *processing is restricted*, that their personal data is *amended*, *corrected*, or *erased*, that they *receive a copy* of their data, and that *data flows are suspended*. In addition to the effect that an intervenability requirement shall have, it has a type describing how data subjects or supervisory authorities can cause the wanted effects. As these

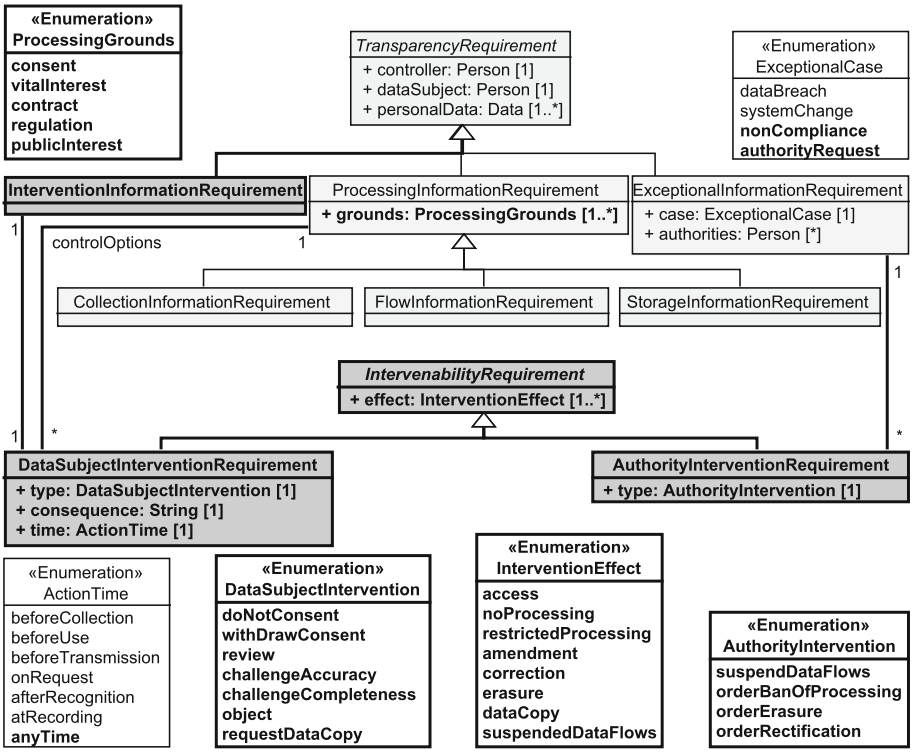


Fig. 2. Our combined taxonomy of transparency and intervenability requirements.

Table 2. Mapping between taxonomy and preliminary requirements

Requirement	Attribute	In/Tn
IntervenabilityRequirement	effect	I1, I3, I5, I7, I8, I10–I13, I15–I18
DataSubjectInterventionRequirement	type	I1–I5, I7, I8, I10–I15
	time	I6, I9, I14
	consequences	T1, T3, I6
AuthorityInterventionRequirement	type	I16, I17, I18
ProcessingInformationRequirement	controlOptions	T1, T3, I6
	grounds	T2
ExceptionalInformationRequirement	exceptionalCase	I16, I17, I18, T7
InterventionInformationRequirement		T4, T5, T6

Table 3. Mapping between authority intervention types and intervention effects

Intervention Type	Possible Intervention Effects	Source
suspendDataFlows	suspendedDataFlows	I18
orderBanOfProcessing	noProcessing, restrictedProcessing	I17
orderErasure	erasure	I16
orderRectification	correction, amendment	I16

types differ for data subjects and authorities, we added the attribute type to the intervenability requirements DataSubjectInterventionRequirement (representing intervention possibilities for data subjects) and AuthorityInterventionRequirement (representing intervention possibilities for supervisory authorities).

AuthorityInterventionRequirement. Almost all initial requirements describe rights of data subjects to influence how their personal data is processed. Only I16, I17, I18, and T7 present possibilities for supervisory authorities to intervene in the processing of personal data. The intervention types for authorities are summarized in the enumeration AuthorityIntervention (cf. Fig. 2). Supervisory authorities may order to *suspend data flows*, order a *ban of processing* of personal data, and order the *erasure* or *rectification* of personal data. The initial requirements I16, I17, and I18 also describe which type of intervention shall lead to which kind of intervention effect. Hence, there are limitations for the combination of intervention types and effects when an ExceptionalInformationRequirement is instantiated. Table 3 presents the valid combinations of intervention types and effects.

T7 indicates that supervisory authorities have to be informed about the processing in order to exercise their rights to intervention properly. Hence, each AuthorityInterventionRequirement has an ExceptionalInformationRequirement assigned that describes which supervisory authorities may intervene. We newly introduced into the enumeration ExceptionalCase the literals nonCompliance and authorityRequest to reflect that authorities have to be informed in the case of processing of personal data in a way that does *not comply* with the regulations and that authorities then have the possibility to intervene in this processing. Additionally, authorities have the right to *request* information concerning the processing of personal data from the controller.

DataSubjectInterventionRequirement. The DataSubjectInterventionRequirement presents the possibilities for data subjects to intervene in the processing of their personal data. These possibilities are summarized in the enumeration DataSubjectIntervention (cf. Fig. 2) that we derived from the preliminary requirements I1–I5, I7, I8, and I10–I15. These initial requirements additionally describe which combinations of intervention types and effects are allowed for DataSubjectInterventionRequirements. The valid combinations are shown in Table 4.

T1, T3, I6, and I9 require that data subjects have to be informed about how they can intervene in the processing of their personal data. To reflect this, we introduced the association controlOptions between DataSubjectInterventionRequirement and ProcessingInformationRequirement (cf. Fig. 2). From the

Table 4. Mapping between data subject intervention types and intervention effects

Intervention Type	Possible Intervention Effects	Source
doNotConsent	noProcessing	I2
withDrawConsent	noProcessing, restrictedProcessing, erasure	I4, I5, I10, I12
review	access	I7
challengeAccuracy	correction, amendment, erasure	I8
challengeCompleteness	amendment, erasure	I8
object	noProcessing, restrictedProcessing, erasure	I10, I12, I15
requestDataCopy	dataCopy	I13

perspective of the ProcessingInformationRequirement, the association describes which options exist for data subjects to intervene in the processing of their personal data. The two attributes consequence and time of DataSubjectInterventionRequirement are used to describe further details on the control option described by the DataSubjectInterventionRequirement. The attribute consequences allows to provide a textual description of the consequences that the utilization of the corresponding intervenability option has. The attribute time describes when data subjects can exercise the corresponding option.

From the preliminary requirements T4–T6, we identified that an additional transparency requirement should be added to the taxonomy. This requirement states the need to inform data subjects about the progress or rejection of interventions requested by them. For this purpose, we introduce the InterventionInformationRequirement. Each DataSubjectInterventionRequirement is associated to an InterventionInformationRequirement and vice versa that presents the need to inform data subjects about the progress or rejection of their intervention.

Furthermore, we identified from T2 that the ProcessingInformationRequirement should also inform data subjects about the legal grounds on which their data is processed. For this, we enriched this requirement with an attribute grounds that reflects the possible grounds for processing personal data by the controller. These are derived from ISO 29100 and the draft of the EU data protection regulation. They are *consent* of the data subject, the *vital interest* of the data subject, an existing *contract*, a *regulation* that allows the processing, and *public interest*.

3 Validation of the Taxonomy Using Related Literature

In this section, we give an overview of existing research that also contains considerations about the privacy goal of intervenability. To validate our proposed taxonomy, we map the notions and concepts used in the related literature to our taxonomy to check whether it is suitable to reflect the intervenability concepts used in the literature.

To identify the relevant related work, we performed a systematic literature review using backward snowballing [11]. To obtain the starting set of papers for

our review, we manually searched the proceedings and issues of the last 10 years of computer science conferences and journals that are mainly concerned with at least one of the topics privacy, requirements, and software engineering and ranked at least as *B-level* in the CORE2014[2] ranking. In this way, we selected 15 conferences and 19 journals. First, we checked whether title or abstract of a paper indicates that the paper is concerned with privacy (requirements), intervenability, empowerment, user's controls, or user's choices. In this way, we obtained 219 articles. We then analyzed the full texts of these articles. Doing this, we reduced the number of relevant articles to 21. Due to the manual search process, we have to deal with the threat to validity that our starting set of papers does not contain all relevant literature, because it was published in a source that we did not consider or was published earlier than in the last 10 years. To mitigate this threat, we applied backward snowballing. That is, we also considered the papers referenced in the papers that we identified as relevant until no new candidates were found. During the snowballing, we identified 79 possibly relevant articles from which 12 were finally considered as relevant. In total, we identified 298 papers that seemed to be relevant after reading title and abstract. After the analysis of the full text, we finally identified 33 papers as related work. Due to space limitations, we cannot present all details of the literature review in this paper, but we provide an overview of our key findings.

The most important finding is that we are able to map each explicitly mentioned intervenability-related concept in the literature to an element of our taxonomy and that none of the articles provides such a structured overview of intervenability requirements and relates these explicitly to transparency requirements. Table 5 shows to which degree the articles identified during the literature review address the intervenability requirements that we identified in this work. For each article, we investigated to which degree aspects of the DataSubjectInterventionRequirement (column **DIR**), the AuthorityInterventionRequirement (column **AIR**), and the relations between intervenability and transparency requirements (column **RIT**) are mentioned in it. We distinguish in Table 5 three cases. If all aspects are addressed, we denote this with a "+". If the aspects are only partially considered, then we denote this with a "o". If no aspects are addressed, we denote this with a "−".

From Table 5, we can see that no article discusses all aspects concerning the relation between intervenability and transparency requirements. Several papers mention that transparency is a prerequisite for intervenability or that data subjects have to be aware of their options to intervene in the processing of their personal data, but none of the papers mentioned that data subjects have to be informed about the progress of the intervention requests they have triggered. Few of the articles considered the intervention options of supervisory authorities. Only three articles covered all of the aspects and 5 identified the need to be able to answer requests of supervisory authorities in order to prove compliance with regulations or standards. All articles discuss at least partially options for the data subject to intervene into the processing of their personal data. The most

[2] http://www.core.edu.au/conference-portal (accessed on 20 June 2016).

Table 5. Mapping of intervenability notions from the literature to our taxonomy

Source	DIR	AIR	RIT
Bier [12], Hansen [5]	+	+	o
Hoepman [13]	+	o	o
Mouratidis et al. [14]	o	o	o
Miyazaki et al. [15]	+	+	–
Kalloniatis et al. [16,17], Spiekermann and Cranor [18]	o	o	–
Makri and Lambrinoudakis [19], Acquisti et al. [20], Masiello [21], Krol and Preibusch [22], Deng et al. [23], Komanduri et al. [24], Cranor [25], Wicker and Schrader [26]	o	–	o
Strickland and Hunt [27], Sheth et al. [28], Fhom and Bayarou [29], Antón et al. [30,31], Van der Sype and Seigneur [32], Basso et al. [33]	+	–	–
Lobato et al. [34], Caron et al. [35], Zuiderveen Borgesius [36], Breaux [37], Langheinrich [38], Feigenbaum et al. [39], Wright and Raab [40], Guarda and Zannone [41], Hedbom [42], Smith et al. [43]	o	–	–

DIR: DataSubjectInterventionRequirement, AIR: AuthorityInterventionRequirement
RIT: Relation between intervenability and transparency requirements

often discussed intervenability option is to consent or withdraw consent. Another interesting observation that we made is that only Hoepman [13] discusses the right to data portability. This right, its implementation, and consequences seem to not yet have been discussed deeply in the literature.

4 Conclusions

In this paper, (1) we systematically derived requirements for the privacy goal intervenability and related transparency requirements from the ISO 29100 standard [7] and the draft of the EU data protection regulation [8]. (2) We then integrated these requirements into an existing metamodel for transparency requirements [6]. The new metamodel provides an overview of the identified kinds of transparency and intervenability requirements and how these are related to each other. The metamodel shall furthermore help requirements engineers to identify and document the transparency and intervenability requirements relevant for them and the information needed to address the transparency and intervenability requirements. (3) We performed a systematic literature review and provide an overview of the relevant research related to intervenability requirements. (4) We validated that our taxonomy contains all necessary aspects mentioned in the identified literature. The literature review showed that all aspects of the privacy goal intervenability mentioned in the literature are reflected in the proposed taxonomy. Furthermore, we did not find any literature that presents

intervenability requirements and their relation to transparency requirements in such a structured, detailed, and complete manner.

We believe that our taxonomy is flexible enough to also represent intervenability and transparency requirements from other regulations and standards, because our proposed metamodel of the taxonomy can easily be adopted and extended. In these cases our metamodel can be enhanced with, e.g., further intervention types and effects. These can easily be added to the corresponding enumerations (cf. Fig. 2).

For future research, we identified three open research questions. (1) How can the taxonomy be used to derive intervenability requirements for a specific software to be developed? To answer this question, we want to integrate the intervenability requirements and their relations to the transparency requirements into our method for the automatic identification and validation of privacy requirements [44]. (2) Which kinds of threats to transparency and intervenability requirements exist? (3) Which technologies exist that implement transparency and intervenability requirements or mitigate threats to these? To address the latter two questions, we plan to set up a catalog of threats that possibly lead to a violation of the identified transparency and intervenability requirements and related mechanisms that may be used to mitigate the identified threats. Based on this catalog, we want to develop a systematic method to identify the relevant threats for a given set of functional requirements and appropriate countermeasures in order to perform a privacy risk assessment.

Acknowledgment. We thank Sylbie Sabit who provided a starting point for this research with her master thesis [45].

References

1. GSMA: MOBILE PRIVACY: consumer research insights and considerations for policymakers, February 2014. http://www.gsma.com/publicpolicy/wp-content/uploads/2014/02/MOBILE_PRIVACY_Consumer_research_insights_and_considerations_for_policymakers-Final.pdf. Accessed 20 June 2016
2. Symantec: State of Privacy Report 2015 (2015). https://www.symantec.com/content/en/us/about/presskits/b-state-of-privacy-report-2015.pdf. Accessed 20 June 2016
3. Quah, A.M.Y., Röhm, U.: User awareness and policy compliance of data privacy in cloud computing. In: Proceedings of the First Australasian Web Conference, AWC 2013, vol. 144, pp. 3–12, Darlinghurst, Australia, Australian Computer Society, Inc. (2013)
4. Ackerman, M.S., Cranor, L.F., Reagle, J.: Privacy in e-Commerce: examining user scenarios and privacy preferences. In: Proceedings of the 1st ACM Conference on Electronic Commerce, EC 1999, New York, NY, USA, pp. 1–8. ACM (1999)
5. Hansen, M.: Top 10 mistakes in system design from a privacy perspective and privacy protection goals. In: Camenisch, J., Crispo, B., Fischer-Hübner, S., Leenes, R., Russello, G. (eds.) Privacy and Identity Management for Life. IFIP AICT, vol. 375, pp. 14–31. Springer, Heidelberg (2012)

6. Meis, R., Wirtz, R., Heisel, M.: A taxonomy of requirements for the privacy goal transparency. In: Fischer-Hübner, S., Lambrinoudakis, C., López, J. (eds.) TrustBus 2015. LNCS, vol. 9264, pp. 195–209. Springer, Heidelberg (2015)
7. ISO/IEC: ISO/IEC 29100:2011 Information technology - Security techniques - Privacy Framework. Technical report, International Organization for Standardization and International Electrotechnical Commission (2011)
8. European Commission: Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52012PC0011. Accessed 20 June 2016
9. OECD: OECD guidelines on the protection of privacy and transborder flows of personal data. Technical report, Organisation of Economic Co-Operation and Development (1980)
10. US Federal Trade Commission: Privacy online: Fair information practices in the electronic marketplace, a report to congress (2000)
11. Jalali, S., Wohlin, C.: Systematic literature studies: database searches vs. backward snowballing. In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2012, pp. 29–38. ACM (2012)
12. Bier, C.: How usage control and provenance tracking get together - a data protection perspective. In: IEEE Security and Privacy Workshops (SPW), pp. 13–17, May 2013
13. Hoepman, J.: Privacy design strategies - (extended abstract). In: Cuppens-Boulahia, N., Cuppens, F., Jajodia, S., El Kalam, A.A., Sans, T. (eds.) ICT Systems Security and Privacy Protection. IFIP Advances in Information and Communication Technology, vol. 428, pp. 446–459. Springer, Heidelberg (2014)
14. Mouratidis, H., Islam, S., Kalloniatis, C., Gritzalis, S.: A framework to support selection of cloud providers based on security and privacy requirements. J. Syst. Softw. **86**(9), 2276–2293 (2013)
15. Miyazaki, S., Mead, N., Zhan, J.: Computer-aided privacy requirements elicitation technique. In: IEEE Asia-Pacific Services Computing Conference (APSCC), pp. 367–372, December 2008
16. Kalloniatis, C., Mouratidis, H., Vassilis, M., Islam, S., Gritzalis, S., Kavakli, E.: Towards the design of secure and privacy-oriented information systems in the cloud: identifying the major concepts. Comput. Stand. Interfaces **36**(4), 759–775 (2014)
17. Kalloniatis, C.: Designing privacy-aware systems in the cloud. In: Fischer-Hübner, S., Lambrinoudakis, C., López, J. (eds.) TrustBus 2015. LNCS, vol. 9264, pp. 113–123. Springer, Heidelberg (2015)
18. Spiekermann, S., Cranor, L.: Engineering privacy. IEEE Trans. Softw. Eng. **35**(1), 67–82 (2009)
19. Makri, E.-L., Lambrinoudakis, C.: Privacy principles: towards a common privacy audit methodology. In: Fischer-Hübner, S., Lambrinoudakis, C., López, J. (eds.) TrustBus 2015. LNCS, vol. 9264, pp. 219–234. Springer, Heidelberg (2015)
20. Acquisti, A., Adjerid, I., Brandimarte, L.: Gone in 15 seconds: the limits of privacy transparency and control. IEEE Secur. Priv. **11**(4), 72–74 (2013)
21. Masiello, B.: Deconstructing the privacy experience. IEEE Secur. Priv. **7**(4), 68–70 (2009)
22. Krol, K., Preibusch, S.: Effortless privacy negotiations. IEEE Secur. Priv. **13**(3), 88–91 (2015)

23. Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W.: A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. RE **16**, 3–32 (2011)
24. Komanduri, S., Shay, R., Norcie, G., Ur, B., Cranor, L.F.: Adchoices? compliance with online behavioral advertising notice and choice requirements. Technical report, CyLab - Carnegie Mellon University (2011). https://www.cylab.cmu.edu/files/pdfs/tech_reports/CMUCyLab11005.pdf. Accessed 20 June 2016
25. Cranor, L.F.: Necessary but not sufficient: standardized mechanisms for privacy notice and choice. JTHTL **10**(2), 273–308 (2012)
26. Wicker, S., Schrader, D.: Privacy-aware design principles for information networks. Proc. IEEE **99**(2), 330–350 (2011)
27. Strickland, L.S., Hunt, L.E.: Technology, security, and individual privacy: new tools, new threats, and new public perceptions: research articles. J. Am. Soc. Inf. Sci. Technol. **56**(3), 221–234 (2005)
28. Sheth, S., Kaiser, G., Maalej, W.: Us and them: a study of privacy requirements across North America, Asia, and Europe. In: Proceedings of the 36th International Conference on Software Engineering, ICSE 2014, pp. 859–870. ACM (2014)
29. Fhom, H., Bayarou, K.: Towards a holistic privacy engineering approach for smart grid systems. In: IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 234–241, November 2011
30. Antón, A.I., Earp, J.B., Reese, A.: Analyzing website privacy requirements using a privacy goal taxonomy. In: IEEE International Confernce on Requirements Engineering, pp. 23–31 (2002)
31. Antón, A.I.: Earp: a requirements taxonomy for reducing web site privacy vulnerabilities. Requirements Eng. **9**(3), 169–185 (2004)
32. Sype, Y.S.V.D., Seigneur, J.: Case study: legal requirements for the use of social login features for online reputation updates. In: Cho, Y., Shin, S.Y., Kim, S., Hung, C., Hong, J. (eds.) Symposium on Applied Computing, SAC, pp. 1698–1705. ACM (2014)
33. Basso, T., Moraes, R., Jino, M., Vieira, M.: Requirements, design and evaluation of a privacy reference architecture for web applications and services. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 1425–1432. ACM (2015)
34. Lobato, L., Fernandez, E., Zorzo, S.: Patterns to support the development of privacy policies. In: International Conference on Availability, Reliability and Security (ARES), pp. 744–749, March 2009
35. Caron, X., Bosua, R., Maynard, S.B., Ahmad, A.: The internet of things (iot) and its impact on individual privacy: an Australian perspective. Comput. Law Secur. Rev. **32**(1), 4–15 (2016)
36. Borgesius, F.Z.: Informed consent: we can do better to defend privacy. IEEE Secur. Priv. **13**(2), 103–107 (2015)
37. Breaux, T.: Privacy requirements in an age of increased sharing. IEEE Softw. **31**(5), 24–27 (2014)
38. Langheinrich, M.: Privacy by design — principles of privacy-aware ubiquitous systems. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) UbiComp 2001. LNCS, vol. 2201, pp. 273–291. Springer, Heidelberg (2001)
39. Feigenbaum, J., Freedman, M.J., Sander, T., Shostack, A.: Privacy engineering for digital rights management systems. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, pp. 76–105. Springer, Heidelberg (2002)
40. Wright, D., Raab, C.: Privacy principles, risks and harms. Int. Rev. Law, Comput. Technol. **28**(3), 277–298 (2014)

41. Guarda, P., Zannone, N.: Towards the development of privacy-aware systems. Inf. Softw. Technol. **51**(2), 337–350 (2009)
42. Hedbom, H.: A survey on transparency tools for enhancing privacy. In: Matyáš, V., Fischer-Hübner, S., Cvrček, D., Švenda, P. (eds.) The Future of Identity. IFIP AICT, vol. 298, pp. 67–82. Springer, Heidelberg (2009)
43. Smith, H.J., Dinev, T., Xu, H.: Information privacy research: an interdisciplinary review. MIS Q. **35**(4), 989–1016 (2011)
44. Meis, R., Heisel, M.: Computer-aided identification and validation of privacy requirements. Information **7**(2), 28 (2016)
45. Sabit, S.: Consideration of intervenability requirements in software development. Master thesis, University of Duisburg-Essen, Germany, August 2015

Information Audit and Trust

Design of a Log Management Infrastructure Using Meta-Network Analysis

Vasileios Anastopoulos[1] and Sokratis Katsikas[1,2(✉)]

[1] Systems Security Laboratory, Department of Digital Systems, University of Piraeus, 18532 Piraeus, Greece
vasanasto@gmail.com, ska@unipi.gr
[2] Center for Cyber and Information Security, Norwegian University of Science and Technology, Gjøvik 2802, Norway
sokratis.katsikas@ntnu.no

Abstract. The need for compliance or organization specific requirements is often guiding the implementation of a log management infrastructure. On a large scale infrastructure the log data are stored in various places, where analysts or administrators need to perform specific analysis tasks. In this work we propose a method for validating the design of the log collector part of the infrastructure, ensuring that each log collector has at its disposal the necessary log data for performing the desired analysis tasks. This is achieved by modeling the infrastructure as an organization and by applying social network analysis concepts and metrics that are used to analyze the structure and performance of real organizations. An example case study, demonstrating the workings of the method and the interpretation of the results, on a simulated infrastructure is also presented.

Keywords: Log management · Social network analysis · Organizational risk analyzer

1 Introduction

According to [1], "a log management infrastructure consists of the hardware, software, networks, and media used to generate, transmit, store, analyze, and dispose of log data". A log management infrastructure typically comprises three tiers, namely the *Log Generation* tier that contains the hosts that generate the log data; the *Log Analysis and Storage* tier, which is composed of one or more log servers, often called *collectors* or *aggregators*, that receive log data or copies of log data from the hosts in the first tier; and the *Log Monitoring* tier, that contains consoles that may be used to monitor and review log data and the results of automated analysis [1–3].

Designing a log management infrastructure for a Wide Area Network (WAN) is a demanding task, especially when the infrastructure is composed of geographically dispersed and heterogeneous networks. The problem underlying most log management related challenges is the need to effectively balance the available amount of log management resources with the ever-increasing supply of log data, generated by various devices throughout the organization, in inconsistent log formats and log

S. Katsikas et al. (Eds.): TrustBus 2016, LNCS 9830, pp. 97–110, 2016.
DOI: 10.1007/978-3-319-44341-6_7

content [1]. Typically in an infrastructure the data generated from various devices of interest are sent to central points of collection, implementing a hierarchy of log collectors. Selecting a centrally managed or distributed architecture is a common practice depending on the needs of or the limitations posed to the organization that owns or operates the WAN. On the other hand, a log management infrastructure has to satisfy the log management requirements for which it was built. The accomplishment of related tasks (e.g. generate a report by tracking user login patterns, identify failed access attempts, software installation, data exfiltration, etc.) requires the availability of the adequate log data at the right time and in the right place. Depending on the requirements or the infrastructure design, specific log data may be stored into specific log collectors and different log management tasks may be performed at different places. In [1] the implementation of multiple log servers, each performing specific analysis or storage functions for specific log generators, was proposed as part of a typical architectural solution. On a large scale infrastructure, each log collector can store a portion of the total generated log files, and each analyst will likely need to perform specific analysis tasks in different places.

The placement of the log collectors within the infrastructure is a design problem that is commonly resolved empirically, taking into account good practices and relevant guidelines. For example, the placement of log collectors needs to take into account the following criteria [3]:

1. **Geographic location:** For a WAN that extends across multiple locations the placement of a collection point to each region is proposed.
2. **Collectors close to their originators:** The collectors have to be placed close to their originators. This is desired as a network problem or the spread of a malware infection could disrupt the log collection process.
3. **Hub-and-spoke architecture**: Many originators forward the log data to a collector and many collectors forward them to a central point or in the case of multiple layers to a central collection point at the following layer.
4. **Hierarchical:** The placement of the components has to follow a hierarchical fashion starting from the originators and ending to the central collection point, where all the log data are collected and processed by the organization.

Thus, the driving factors for making design decisions on the placement of log collectors do not take into account the log analysis tasks that will need to be performed using the log data stored in each collector. Hence, there is a need to validate the design structure of the second tier of a log management infrastructure.

This design structure comprises the relationship among collectors, log management tasks and log data. Validation here means ensuring that on each log collector the desired log management tasks can be performed using the log data actually collected on it on one hand and establishing the significance of each log collector for the accomplishment of the overall log management tasks on the other hand. The entities and relationships forming the design structure of the log management infrastructure can be modeled as a collection of interlocked networks called a Meta-Network. A Meta-Network can be represented using the Meta-Matrix conceptual framework [5]; this allows the extension of the Social Network Analysis (SNA) concepts and techniques [6] to those of the Meta-Network Analysis (MNA) [7].

In this paper we consider the log management infrastructure to be a complex organization. This viewpoint enables the application of concepts and measures used to analyze the performance and structural properties of real organizations, such as the MNA. As a result, we propose a method for validating the design and configuration of a large scale log management infrastructure and for establishing a measure of each log collector's criticality for the infrastructure. This allows taking corrective action as well as the continuous evaluation of the performance of the log management infrastructure.

The remainder of this paper is organized as follows: In Sect. 2, the related work is presented. In Sect. 3, we elaborate on the proposed approach. Section 4 reports on the findings of applying the proposed approach to a simulated log management infrastructure. Our conclusions and future work are discussed in Sect. 5.

2 Related Work

A high-level viewpoint of log management technologies is given in [1]. This paper can be used as a framework for the development, the implementation and the maintenance of a log management infrastructure, as it addresses most issues at a high level. In [2] the authors propose a log data collection framework that is composed of four steps. It starts with the definition of the potential threats posed to the organization, which are then prioritized based on their risk. It continues with the identification of the data feeds that are necessary to address each technical threat, and concludes with the detailed analysis of the selected data sources. A high level guide is also available in [4], composed of five phases for building a log analysis system. The planning phase, where the system requirements are specified, the software selection, where issues related to the selection of the right software tools are discussed, the policy definition, where procedures and routines are defined. The fourth phase guides the decision for the system architecture and at the last phase, issues related to the scaling of the system are considered. A data acquisition and a data transmission module are presented in [8], as components of a network security monitoring solution, focusing on specific tasks of log management that are required by the proposed system. A log management architecture is proposed in [9], discussing functions related to log collection and storage, and in conjunction with commercial SIEM systems. The authors in [10] propose a method that guarantees the completeness of logs that are transferred through an untrusted network. They focus on the forensic soundness of the transferred log data, they propose a new log format and they discuss functions related to log collection and storage. In [11, 12] the authors present their systems for the delegation of log management to the cloud, achieving properties of data such as availability privacy and confidentiality. These works are not implementation guides but they are indicative of the advances in the employment of cloud solutions for log management tasks and the challenges they pose.

The aforementioned work cannot be used as a step-by-step guide, as it is either abstract or focused on specific issues of log management and relevant systems. The methodology proposed in [13] adopts and adjusts specific elements of previous works to avoid "re-inventing the wheel" whenever it is possible, and differentiates from it as it extends to both high-level and low-level aspects of log management, to both business and technical issues and can guide the design, implementation and evaluation of a log

management infrastructure. The methodology consists of 11 steps and results in the design and configuration of the infrastructure, including the specific log files that are sent to each log collector.

Social network analysis is based on an assumption of the importance of relationships among interacting units. The social network perspective encompasses theories, models, and applications that are expressed in terms of relational concepts or processes. In social network analysis the identification of the key nodes is a task that is commonly achieved using the measurements of centrality. In [6] various measurements are defined along with their possible interpretation and meaning, depending on the context. In [14] various methods of analyzing social networks are presented. One of them is the separation of the social network into a core and a periphery part based on the centrality of the nodes. More complex methods are proposed in [15, 16], where the author addresses the inefficiency of the centrality measures in identifying important and key nodes. SNA usually handles networks with one type of nodes, such as agent networks or task networks. The SNA methods do not lend themselves well to treating complicated data structures such as those encountered in multi-mode networks, where three or more modes may coexist [6]. Therefore, whereas SNA can be used to model and analyze the placement of log collectors, it does not lend itself to modeling and analyzing their design structure, which is the relationship among different types of nodes; collectors, log management tasks and log data. To achieve this, it is essential to extend SNA to multiple types of nodes and to more complex cross-connected networks; this is accomplished by MNA.

Meta-networks were first described as the precedence, commitment of resources, assignment, networks, and skills (PCANS) model [17]. They involve key entities that influence organizational design, such as tasks, resources, knowledge, and agents, as well as their relations [18] and has been applied to diverse fields [7, 19–21]. To our knowledge there is no previous work on using multi-relational social network analysis on log management infrastructures to assess the alignment of the structure design with the relevant log management requirements.

3 Validating the Log Management Infrastructure Design Structure

3.1 Modeling a Log Management Infrastructure as a Social Network

The log management infrastructure comprises three entities, namely the *log management tasks* (component of the requirements); the *log collectors*; and the *log files* and their relationships. Each log management task needs the data contained into specific log files, while each log file is sent to one or more log collectors [13]. Each log collector is "assigned" specific log management tasks, meaning that an analyst, using the data collected at this collector has to be able to perform the defined subset of tasks (or all of them). This results into relationships among the three entities as depicted in Fig. 1; a log file (shown as a square) is linked with log collectors (shown as circles), in a one-to-many relationship; a log collector is linked with tasks (shown as hexagons), in a one-to-many relationship; a log management task is linked with log files, in a one-to-many relationship.

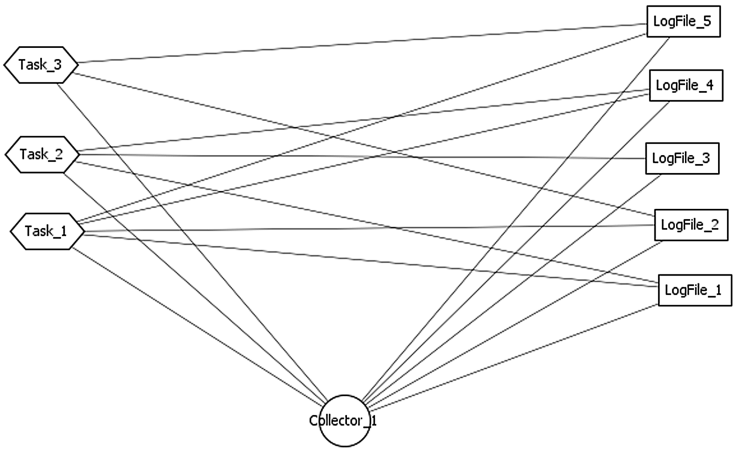


Fig. 1. Relationships among log management infrastructure entities

In SNA a *node* (or actor) is a social entity. It can be a discrete social unit (an individual) or a collective social unit (a group of people, a corporate department, etc.). The term "actor" does not imply that actors have the ability to act. *Links* (or social ties) connect actors, establishing a tie between a pair of actors. A *relation* is the collection of a specific kind of ties formed among the actors within a specific set of actors. *Social networks* are composed of nodes and links. These nodes relate with other nodes through their links. When the links have a direction the network is directed and the link from node A to B is different from node B to A. When a link direction is not specified, the network is undirected and the link from A to B is not different from the one from B to A. A node can have one or more attributes and a link can be binary of valued. Using graph theory notation G = (V, E) is a social network G, with |V| nodes and |E| links among them. It is represented by a |V| × |V| adjacency matrix, where the existence of a link between node $v_i \in V$ and node $v_j \in V$, is indicated by a value in the $e_{ij} \in E$ cell. This is a *one-mode* network [22], as the links are formed among the nodes of the same set, where the term *mode* refers to the distinct set of entities on which structural variables are measured. A *two-mode* network is formed between two distinct sets of nodes, N and M, and is represented by a |N| × |M| incidence matrix. In our case, the identified entities of the log management infrastructure and the relationships among them are used to construct a three-mode social network, whose elements are as follows:

$T = \{t_1, t_2, \ldots, t_r\}$, the log management tasks.
$C = \{c_1, c_2, \ldots, c_c\}$, the log collectors.
$F = \{f_1, f_2, \ldots, f_f\}$, the log files.

and the links (relationships) among them are represented by the following incidence matrices:

|F| × |T|, the log files necessary to perform each log management task.
|F| × |C|, the collector to which each log file is sent.
|T| × |C|, the log management tasks "assigned" to each log collector.

3.2 Modeling the Log Management Infrastructure as a Meta-Network

In the context of MNA, the design structure is composed of *agents*, an information processing entity; *tasks*, part of a set of actions which accomplish an assignment; and *knowledge*, the available information [23]. Each of these entities corresponds to a *node class* and collections of nodes belonging to these classes form specific *nodesets*. Links can be present between the nodes of the same node class or between the nodes of different node classes. Nodes and links may have attributes further describing the nodes and providing context to their relationship. Multiple networks can be created, each one representing a specific type of connection between the nodes. When the network is formed between nodes of the same node set it is a one-mode network, while when it uses N node sets it is an N-mode network. A group of such networks is referred to as a meta-network.

Table 1. Constructed meta-network

Meta-network (organization):		Log management infrastructure
Node class	Node set	Interpretation
Agent (A)	Log collectors (C)	The systems where the log files are collected
Knowledge (K)	Log files (F)	The log files sent by the log sources
Task (T)	Log management tasks (T)	The log management tasks (components of the requirements)
Network	2-mode network	Interpretation
Agent × Task (AT)	|T| × |C|	The log management tasks expected to be accomplished on each log collector
Knowledge × Task (KT')	|F| × |T|	The log files that are necessary to perform each log management task
Agent × Knowledge (AK)	|F| × |C	The log files actually collected on each log collector

In analogy, as shown in Table 1, we consider the log management infrastructure as an organization composed of log collectors (agents), log files (knowledge) and log management tasks (tasks). Further, following the notation of [23], AT is the *agent* x *task* matrix, KT′ is the transposed matrix *knowledge* x *task*, AK the *agent* x *knowledge* matrix; these three networks form the meta-network to be analyzed in order to identify flaws in the design of the infrastructure. It should be noticed that the log files are modeled as knowledge, thus they are independent of life-cycle issues, such as on-line/off-line storage, rotation, compression, etc.

The analysis is based on the following measures [23, 24]:

Agent Knowledge Needs Congruence, is the amount of knowledge that an agent lacks to complete its assigned tasks, expressed as a fraction of the total knowledge required for the assigned tasks. The measure compares the knowledge needs of the agent to do its assigned tasks with the actual knowledge of the agent. The measure value for an agent increases when it has need of knowledge to which it is not assigned. Let NK = AT*KT' be the knowledge needed by agents to do their assigned tasks; then the output value for agent *i* is sum(NK(i,:) .* ∼AK(i,:))/sum(NK(i,:)).

Agent Knowledge Waste Congruence, is the amount of knowledge that an agent has that is not needed by any of its tasks expressed as a fraction of the total knowledge of the agent. The formula compares the knowledge of the agent with the knowledge it actually needs to do its tasks. Any unused knowledge is considered wasted. Let NK = AT*KT' be the knowledge needed by agents to do their assigned tasks, then the output value for agent *i* equals to sum(∼NK(i,:) .* AK(i,:))/sum(NK(i,:)).

Knowledge Potential Workload, is the maximum amount of knowledge an agent could use to do tasks if it were assigned to all tasks. If an agent is assigned all the tasks this measure will compute a value expressing its potential to carry out all the tasks based on his connections to the knowledge needed for the tasks [25]. The value for agent *i* equals to sum((AK*KT(i,:))/sum(KT). The higher the value of this measurement, the more tasks can be completed using the knowledge of this node, thus the more critical this node is.

3.3 Validating and Improving the Design Structure

By calculating the first two measures, the log collectors that need more log data in order to accomplish their tasks are identified, as well as the nodes that collect more log data than they actually need. The log sources can then be reconfigured, modifying either the collectors to which their log files are sent, or the specific log files that are sent to each log collector. Following these adjustments, each log collector ends up with the log data it actually needs to perform the "assigned" tasks, thus avoiding the waste of resources. The third measure is used to identify the log collectors that are important for the infrastructure. A collector whose log data are used for a large subset of tasks is more important than a collector used to perform only a few ones, categorizing it as critical for the log management infrastructure.

4 An Example Case Study

In order to demonstrate the workings of the proposed method, we assume a log management infrastructure where 25 log files from various devices are sent to 5 log collectors. The collected log data were used to perform a set of 10 analysis tasks and on each log collector a subset of analysis tasks was desired to be accomplished. The number of nodes and links was selected to be small in this example in order to ensure the readability of the visualizations. Each log analysis task needs specific log files in order to be accomplished. We note, however, that the available MNA tools can easily handle thousands of nodes and links, posing in practice no limit to the scalability of the proposed approach. The log collector for each log file was configured during the implementation of the infrastructure, as well as the subset of analysis tasks for each collector and the log files that are required for each task. Each log collector is placed on a different physical location and it is configured to collect log data from a specific category of devices or operating systems [1]. Four log collectors, one for Linux generated logs; one for Windows generated logs; one for log generated from network devices; one for logs generated from security devices form a layer of collectors,Layer-2. The fifth

collector, Layer-1, receives all the logs from Layer-2 as well as the logs generated by specific services/applications. The aim of the analysis is to validate whether or not the tasks assigned to each collector can be actually performed with the specific log data they collect.

The infrastructure is modeled as a meta-network composed of three node classes and three 2-mode networks. The node classes are agent (A), knowledge (K) and task (T), having the corresponding node sets of log collectors |C| = 5, log files |F| = 25 and log management tasks |T| = 10. A summary of the meta-network data is shown in Table 2.

Table 2. Summary of meta-network data

Node class	Node set	Node name
Agent (A)	|C|=5	Layer-1-Central (c1)
		Layer-2-Windows (c2)
		Layer2-Linux (c3)
		Layer-2-Network Devices (c4)
		Layer-2-Security Devices (c5)
Knowledge (K)	|F|=25	Windows-Security log (f1)
		Linux-secure/auth log (f5)
		VPN Server log (f10)
		Firewall log (f12)
		Antivirus log (f19)
Task (T)	|T|=10	Authentication failures and successes (t1)
		Execution of scheduled tasks (t3)
		Outbound connections from internal and DMZ systems (t6)
		Critical errors (t8)
		Malware (t9)

The three 2-mode networks are the log files collected on each log collector |F| × |C|, the log files that are required for each analysis task |F| × |T| and the analysis tasks "assigned" to each collector |T| × |C|. Sample data of the matrices representing the aforementioned networks are listed in Tables 3, 4 and 5, respectively.

For the needs of this case study we assume three teams of personnel in the organization; the security, the system administration and the network administration teams. Each team is located in a different physical location and has access to log collectors as follows:

- Security team: Layer-2-Security Devices (c5)
- Systems' administration team: Layer-2-Windows (c2), Layer-2-Linux (c3)
- Network administration team: Layer-2-Network Devices (c4)

Each team needs to perform a subset of analysis tasks as shown in Table 5.

When these 2-mode networks are combined, the resulting 3-mode network is visualized in Fig. 2. In this figure, the circles represent the log collectors, the hexagons the analysis tasks and the squares the log files.

Table 3. Sample log_file × collector matrix

Collector	Windows-security log (f1)	Windows-scheduled tasks log (f2)	Windows-system log (f3)	Linux-secure/auth log (f5)	Firewall log (f12)
c1	1	1	1	1	1
c2	1	1	1	0	0
c3	0	0	0	1	0
c4	0	0	0	0	0
c5	0	0	0	0	1

Table 4. Sample log_file × analysis_task matrix

Task	Windows-security log (f1)	Windows-scheduled tasks log (f2)	Windows-system log (f3)	Linux-secure/auth log (f5)	Firewall log (f12)
t1	1	0	0	1	0
t3	0	1	0	0	0
t6	0	0	0	0	1
t8	0	0	1	0	0
t9	1	0	0	1	0

The construction of the multi-mode social network, the visualizations and the measurements were performed using the CASOS ORA-NetScenes 3.0.9.9.29 tool [26], developed by Carnegie Mellon University. It is a network analysis tool used to detect risks or vulnerabilities on the design structure of organizations, analyzing their structural properties. The calculated measures for this case study are shown in Table 6. The *Agent Knowledge Waste Congruence* of *c1* is one, meaning that the log data collected on this collector is not needed for the analysis.

This was expected, as this collector is the central point where every log file is stored, though in Table 5 we observe that no analysis task is "assigned" to it. Collector *c4* has a value of 0.500 as it receives both syslog and NetFlow protocol data, though it only needs the syslog data for the "assigned" tasks. Concerning the *Agent Knowledge Needs Congruence,* we observe that no collector has the required log data. The value for *c4* is the highest, as the syslog data it receives from the network devices are not enough to track the system and service restarts, the critical error, etc. throughout the infrastructure.

Applying MNA on this simulated infrastructure we were able to quickly identify that log collector *c4* not only lacks the necessary log data to perform its tasks, but it receives log data irrelevant to the "assigned" tasks as well; this results in waste of resources. None of the remaining collectors has at its disposal the necessary data, but they avoid receiving irrelevant log data, too. Figure 3 visualizes the c4 node and its links. The dotted lines represent the log files it actually receives and the solid lines represent the log files it should be receiving to accomplish its tasks. The social network's layout has been adjusted for readability and the positions of the nodes have no special meaning. As a result of the analysis, a collector that both lacked necessary and received unnecessary information was identified, and proper adjustments can be easily performed assisted by the visualizations. This analysis should be repeated for collectors

Table 5. Analysis_task × collector matrix

Collector	Authentication failures and successes (t1)	Multiple login failures followed by success (t2)	Execution of scheduled tasks (t3)	System and service restarts and shutdowns (t4)	Application install and updates (t5)	Outbound connections from internal and DMZ systems (t6)	File transfers (t7)	Critical errors (f8)	Malware (f9)	Database users executing CREATE, GRANT (f10)
c1	0	0	0	0	0	0	0	0	0	0
c2	1	0	1	1	1	0	1	1	0	0
c3	1	0	1	1	1	0	1	1	0	0
c4	1	0	0	1	0	0	0	1	0	0
c5	1	1	1	1	1	1	1	0	1	1

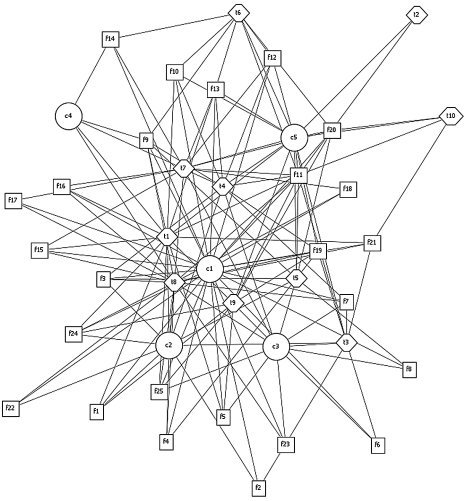


Fig. 2. Generated 3-mode social network

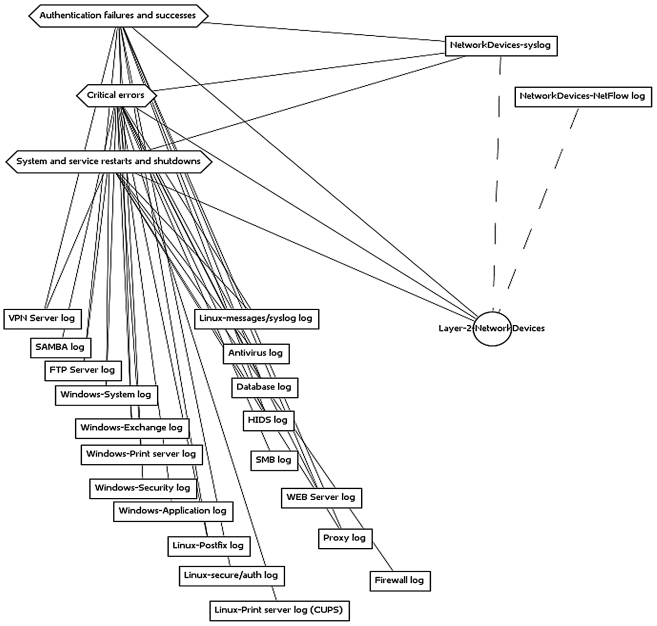


Fig. 3. Links of the *c4* node

c2, c3 and *c5*, as well as the process of adjustments and analysis until the desired measurements are achieved.

Moving to the identification of nodes' criticality, ignoring the central collector (*c1),* the most important collectors for the performance of all the log management analysis tasks are, in decreasing importance, *c5, c2, c3* and *c4,* based on the measurement of

Table 6. MNA measurements

Collector	Congruence, agent knowledge waste	Congruence, agent knowledge needs	Potential workload, knowledge
c1	1.000	0	1
c2	0	0.821	0.162
c3	0	0.839	0.149
c4	0.500	0.912	0.095
c5	0	0.525	0.405

the *Knowledge Potential Workload.* Following this analysis the files that are missing or are in surplus on each collector can be easily identified, and corrective actions may be applied, by reconfiguring the log files that are sent to each log collector.

Apart from having predefined subsets of analysis tasks "assigned" to each collector, a different use case could be that of an analyst seeking the optimal collectors in order to perform specific analysis tasks as part of their investigation, be it related to security or not.

5 Conclusions and Future Work

In this work we considered a large scale log management infrastructure as a complex organization. This allowed us to model it as a multi-mode social network formed by the links among its components, namely the log management tasks, the log files and the log collectors. The MNA measures used for the structural analysis of organizations were applied on the generated social network, aiming to identify flaws in the design of the infrastructure and to enable the application of corrective actions. We target specifically in identifying log collectors lacking the necessary log data that are required to perform the desired log management tasks, as well as to categorize the log collectors in terms of criticality for the infrastructure. The proposed method has been successfully applied on a small-scale simulated infrastructure. However, even open source MNA software can easily handle thousands of nodes and links rendering the proposed method to be highly scalable. The proposed method focused on the log collectors; however, the analysis could also include the log files and the log analysis tasks. Future work could study the importance of each log file in fulfilling the requirements, as well as identify the log analysis tasks that are "demanding" in log files.

References

1. Kent, K., Souppaya, M.: Guide to Computer Security Log Management. NIST SP800-92 (2006). http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf. Accessed 23 Mar 2016
2. Smith, J., Sanders, C.: Applied Network Security Monitoring, 1st edn. Syngress, Oxford (2014)

3. Cisco: Building Scalable Syslog Management Solutions (2015). http://www.cisco.com/c/en/us/products/collateral/services/high-availability/white_paper_c11-557812.html. Accessed 23 Mar 2016
4. Schmidt, K.J., Chuvakin, A.: Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management, 1st edn. Syngress, Oxford (2013)
5. Carley, K.M., Reminga, J.: ORA: Organization Risk Analyzer. CASOS Technical report CMU-ISRI-04-106 (2004)
6. Faust, K., Wasserman, S.: Social Network Analysis: Methods and Applications. Cambridge University Press, Cambridge (1994)
7. Li, Y., Lu, Y., Li, D., Ma, L.: Metanetwork analysis for project task assignment. J. Constr. Eng. Manag. 141(12): (2015). http://dx.doi.org/10.1061/(ASCE)CO.1943-7862.0001019
8. Liu, J., Guo, J., An, R., Gao, K.: Study on data acquisition solution of network security monitoring system. In: 2010 IEEE International Conference on Information Theory and Information Security (ICITIS), Beijing, pp. 674–677 (2010)
9. Rezayi, S., Gharaee, H., Madani, A.: Log management comprehensive architecture in Security Operation Center (SOC). In: International Conference on Computational Aspects of Social Networks (CASoN), Salamanca, pp. 284–289 (2011)
10. Uehara, M., Shimada, Y., Tomono, A.: Trusted log management system (chap. 5). In: Khalil, I., Mantoro, T. (eds.) Trustworthy Ubiquitous Computing, pp. 79–98. Springer, Atlantis Press, Berlin (2012)
11. Kala, T.K., Murugan, A.: An effective secured cloud based log management system using homomorphic encryption. Int. J. Comput. Sci. Inf. Technol. **5**(2), 2268–2271 (2014)
12. PawarAnil, S., RajebhosaleSagar, B.: Development of highly secured cloud rendered log management system. Int. J. Comput. Appl. **108**(16), December 2014
13. Anastopoulos, V., Katsikas, S.: A methodology for building a log management infrastructure. In: Proceedings of IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2014), pp. 301–306 (2015). doi:10.1109/ISSPIT.2014.7300604
14. Mrvar, A., Batagelj, V., Nooy, W.D.: Exploratory Social Network Analysis with Pajek (Structural Analysis in the Social Sciences), 2nd edn. Cambridge University Press, Cambridge (2011)
15. Borgatti, S.P.: The key player problem. In: Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers (2003)
16. Borgatti, S.: Identifying Sets of Key Players in a Social Network, pp. 21–34. Springer Science, Berlin (2006)
17. Krackhardt, D., Carley, K.M.: PCANS Model of Structure in Organizations. Institute for Complex Engineered Systems, Carnegie Mellon University, Pittsburgh (1998)
18. Carley, K.M.: Computational organizational science and organizational engineering. Simul. Model. Pract. Theor. **10**(5), 253–269 (2002)
19. Wakolbinger, T., Nagurney, A.: Dynamic supernetworks for the integration of social networks and supply chains with electronic commerce: modeling and analysis of buyer-seller relationships with computations. NETNOMICS: Econ. Res. Electron. Netw. **6**(2), 153–185 (2004)
20. Nagurney, A., Wakolbinger, T., Zhao, L.: The evolution and emergence of integrated social and financial networks with electronic transactions: a dynamic supernetwork theory for the modeling, analysis, and computation of financial flows and relationship levels. Comput. Econ. **27**(2–3), 353–393 (2006)
21. Nagurney, A., Dong, J.: Management of knowledge intensive systems as supernetworks: modeling, analysis, computations, and applications. Math. Comput. Model. **42**(3), 397–417 (2005)

22. Tucker, L.R.: Implications of factor analysis of thee-way matrices for measurement of change. In: Harris, C.W. (ed.) Problems in Measuring Change, pp. 122–137. University of Wisconsin Press, Madison (1963)
23. Carley, K.M., Pfeffer, J., Reminga, J., Storrick, J., Columbus, D.: ORA User's Guide 2013. CMU-ISR-13-108, School of Computer Science, Institute for Software Research, Carnegie Mellon University, Pittsburgh, PA 15213, June 2013
24. Lee, J.-S., Carley, K.M.: OrgAhead: a computational model of organizational learning and decision making. Technical report CMU-ISRI-04-117, School of Computer Science, Institute for Software Research International, Carnegie Mellon University, Pittsburgh (2004)
25. Carley, K.M.: Summary of Key Network Measures for Characterizing Organizational Architectures. Carnegie Mellon University, Pittsburgh (2002). Collins, M.S.: Network Security Through Data Analysis: Building Situational Awareness, 1st edn. O'Reilly Media, Sebastopol (2014)
26. Homepage|CASOS. http://www.casos.cs.cmu.edu/index.php. Accessed 23 Mar 2016

The Far Side of Mobile Application Integrated Development Environments

Christos Lyvas[1(✉)], Nikolaos Pitropakis[2], and Costas Lambrinoudakis[1]

[1] Department of Digital Systems, University of Piraeus, Piraeus, Greece
{clyvas,clam}@unipi.gr
[2] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia
pitropakis@gatech.edu

Abstract. Smart phones are, nowadays, a necessity for the vast majority of individuals around the globe. In addition to the ubiquitous computing paradigm supported by such devices, there are numerous software applications that utilize the high computational capabilities that they offer. This type of software is a vital part of what is known as e-Commerce, with a variety of business models proposed and implemented. Lately, a new era of free-ware mobile application has arisen with paid features and promoted content in them. Piracy is not only the weakest point of software's financial ecosystem for conventional computing systems but also for smartphones. Actions like replication, redistribution and licensing violations can cause financial losses of colossal extent to their creators. Mobile applications also introduce the following peculiarity: They are distributed through predefined channels (Application Stores) owned by mobile operating system vendors such as Apple, Google and Microsoft. In this research we present several scenarios where cracked and modified applications can be freely used into every non jailbroken iOS device. Moreover it is demonstrated that not even in strict mobile environments, such as Apple's, end-users should be considered as trusted entities from application developers by default.

Keywords: Application integrity · Application reverse engineer · Application security

1 Introduction

Ubiquitous computing is certainly a breakthrough. Two decades ago no one could imagine that he would be able to carry in his pocket mini computers with extremely high processing power and capable to provide internet access on demand. In a very short time smartphones have established their position in the mobile phone market and have become the accessory that almost everyone uses constantly either for work or for entertainment.

After Apple launched the first iPhone, Google and Microsoft followed, offering new smartphones and smart devices to the public. Each one of them promised to improve our living quality and has developed software that was advertised as secure and stable. During the last couple of years biometric sensors, such as fingerprint sensor and iris

S. Katsikas et al. (Eds.): TrustBus 2016, LNCS 9830, pp. 111–122, 2016.
DOI: 10.1007/978-3-319-44341-6_8

sensor, were introduced as an extra security level for the protection of the user. However in practice most mobile applications do have bugs or other vulnerabilities that can be exploited by malicious parties in order to harm the user. A very interesting debate for academics and users is the following very simple question "which mobile platform among iOS, Windows Phone and Android is more secure?" Clearly, there is not an easy answer, especially since there are a lot of similarities in terms of the security mechanisms adopted by each platform as all of them follow similar technological paths.

The main objective of this paper is to evaluate the mechanisms that the iPhone operating system features in order to check the trustworthiness of the applications. Cracked or prepackaged applications can run on Android devices by simply modifying the default configuration settings of the mobile phone. This is also true for the Windows Phone, where untrusted applications can be deployed into any developer unlocked Windows Phone using the aid of an application deployment tool running on a PC. For Apple devices the most popular method for executing untrusted applications is the Jailbreak procedure that bypasses the code signature mechanism and instantly voids the guarantee. The IOS's Mandatory Code Signature mechanism aims to ensure that an application can be executed only if its code has been signed by a trusted party [1]. Thus, prior to an application's execution, an internal kernel check verifies that the code loaded into the virtual memory contains a valid signature and can, thus, proceed with the execution [5]. Any modification of a signed executable results in the invalidation of the entire file/application. The Mandatory Code Signature mechanism can prevent cracked applications of being executed on trusted devices (not Jailbroken) while at the same time trusted malformed or malicious applications that change their executable code or behave like droppers [6] cannot execute their payload on non-modified iOS devices since the executable code does not have a valid signature. The Jailbreak procedure disables the kernel code sign check, allowing those devices to run pseudo signed code.

When a developer publishes an application, Apple ensures that the application is fully functional, bug free and that it does not violate Apple's security regulations [4]. Following the evaluation, the application is released in the iOS App Store and Mac iTunes. These applications can execute on any iDevice (iPhone, iPod, iPad) [25] since they have been signed with Apple's s private key. This mechanism, as part of the Mandatory Code Signature scheme explained before, ensures that applications with illegal content or malicious payloads will not be executed on trusted devices.

Moreover, all the executables of the applications published in the App Store are code protected with encrypted segments by Apple (connotation of ARMv7-A and ARMv8-A Mach-O compatible for both 32 and 64 Bit ARM architectures) in order to prevent any reverse engineering and replication attempts. This kind of obfuscation however is not effective during runtime dynamic analysis, and thus an attacker can obtain the unencrypted version of an executable when it is loaded into the memory [21].

In this paper we describe costless methods based on iOS Integrated Development Environment, where any user can overcome the code signature mechanism and execute cracked or prepackaged applications onto new iDevices. Moreover, the impact of this ability is highlighted as it could lead to integrity violation of legitimate applications' transactions, such as in app purchases [18], with significant financial consequences for their creators.

The rest of the paper is organized as follows. Section 2 provides an overview of the related work and a comparison with the presented approach. In Sect. 3 the anatomy of an iOS application and its embedded mechanisms is explained, while Sect. 4 describes the provisioning model for iOS devices. Section 5 introduces practical attacks on paid and free applications. Section 6 presents our thoughts for mitigating the attacks as well as pointers for future work and specifically on how the proposed method can be further extended in order to achieve a more in depth investigation of the iOS platform.

2 Related Work

This research work has emphasized into Apple's iOS security ecosystem since it is undoubtedly one of the stricter mobile platforms. Android and iOS cover 92.95 % of mobile market for the last 4 years with an average of 79.1 % and 13.8 % respectively [26]. Nonetheless, an interesting fact about those mobile platforms, is that iOS users spend much bigger amounts of money to purchase applications or features on them, in comparison to Android users [27].

Despite the fact that the huge percentage of software piracy is happening on Jailbroken devices, there are a lot of threats against applications' integrity onto new iDevices also. By combining a series of weaknesses in the development chain of iOS applications it is clearly demonstrated that the entire business model of Apple's App Store is not only threatened by Jailbreak Development but also it cannot mitigate software piracy.

The majority of research work on the iOS application security model has tried to attack the security mechanisms through remote exploits or local privilege escalation vulnerabilities, using memory corruptions and memory leaks with a variety of methods (Return Oriented Programming, Jump Oriented Programming, Heap Spraying etc.).

Wang et al. in [16] have managed to bypass Apple's App Store review process and publish vulnerable applications, while they propose ways to remotely exploit them based on iOS Framework vulnerabilities. In another paper [15] they propose ways to inject malicious developer-signed applications to non-jailbroken iOS devices by intercepting USB and Wi-Fi connection between iDevices and infected computers. Finally, they claim that infected non Jailbroken devices could act as botnets.

A survey by Zheng et al. [14] evaluates all possible ways through which an application can be distributed to a non Jailbroken iOS device signed with a variety of several different paid certificates (Developer or Enterprise). During their research they develop a framework to identify threats induced by the usage of vulnerable iOS private API (undocumented application programming interfaces) functions. They evaluated 1408 private enterprise applications and they discovered several vulnerabilities and privacy leaks in their payloads. Finally, they claim that non jailbroken iOS devices can run cracked iOS applications if the applications have been signed with valid certificates.

A methodology for repackaging iOS applications executed on new 32Bit iDevices was published by Livitt [22]. Specifically, a developer with an enrolled Developer Account, with an annual cost of $99, can generate provisioning profiles (Certificates) suitable to resign App Store Applications through Apple's Developer Portal [24]. After

performing tests with the tool proposed [23], it was concluded that it was only compatible with 32 Bit iDevices.

The novelty of the work presented in this paper (Table 1) lies on the fact that it demonstrates how someone can use any type of application (freeware or paid) freely on any kind of non Jailbroken iDevice. The above procedure is independent of the iOS version and the user only needs his/her Apple ID. Furthermore, additional ways that allow users to access premium features and bypass applications' additional security checks are discussed, while additional developer features can be unlocked and used for reversing third party applications such as automatic network monitoring, memory allocation debugging and automatic memory leak inspection. Finally, it has been demonstrated that in some app purchase cases it is feasible to bypass the payment by modifying the application's configuration files and accessing premium features by replacing legitimate with arbitrary values. This kind of access into third party application files is possible because they were supposed to run onto a new iDevice, owned by the developer who signs them (signed with developer certificate), for testing purposes.

Table 1. Method comparison

		Rethinking & Repackage iOS Apps [22]	Enpublic Apps [14]	The Far Side Of iOS IDE
Member Account	Apple ID (Free)	✗	✗	✓
	Enrolled Developer or Enterprise	✓	✓	✓
Cracked Paid	Objective-C	✗	✓	✓
	Swift	✗	?	✓
Hooked Free	Objective-C	✓	✗	✓
	Swift	✗	✗	✗
Device Architectures (Compatibility)	32 Bit	✓	?	✓
	64 Bit	?	?	✓
In App Purchases Bypass	Configuration File Modification	✗	✗	✓

3 Anatomy of iOS Application

iOS applications can be downloaded through iTunes for conventional devices (Mac, PC) and via App Store for iDevices (iPhone, iPod, iPad) with an active Apple ID account being necessary in all cases. An iOS application is a Zip archive, containing several folders and files. Every application contains a property list file with information about the downloaded ipa (Apple application archive) [21] file, such us which Apple ID was used for the purchase, the version of the application, date of creation etc. Another folder placed in every ipa archive is the Payload folder which carries the application bundle in app file extension. Every legitimate application container carries several application icons, images and files for the application's user interface. In order an application to run in a non Jailbroken iDevice it must contain a valid property list file placed inside the folder _CodeSignature. This property list contains hashes of every file inside the app container in Base64 format [21]. The property list file named "info" inside the application container carries information about the executable version, the unique name of the application (Bundle ID), URLs for the inter app communication mechanism [2] and the publisher's identifier. The executable file of an application is a connotation of ARMv7-A and ARMv8-A Mach-O executables of the production source code. Any additional extension or plugin of the application is most of the times placed inside the bundle folders. For applications developed with swift framework an additional folder exists into the app container which carries the necessary dynamic libraries for the application's execution. Figure 1 depicts the structure that has been already described.

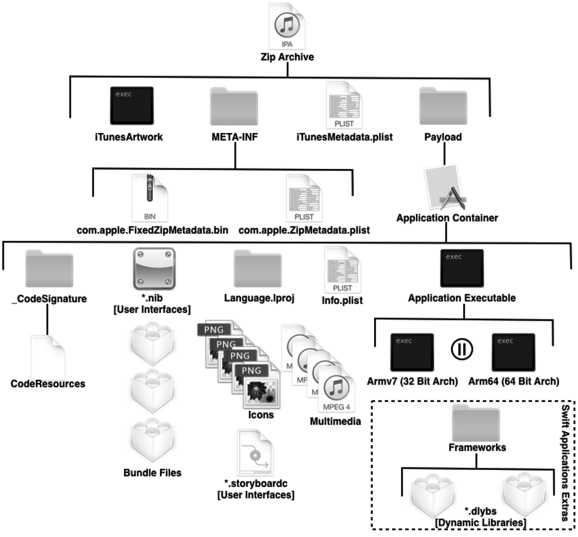


Fig. 1. IPA container

4 Provisioning Profiles

The code sign procedure is based on Public Key Infrastructure implementation which ensures the integrity of the signed objects and the identity of the parties involved. Theoretically, every developer has a pair of public and private RSA 2048 Bit key. As an authority, Apple creates developer certificates based on developers' public keys, then uses the SHA-256 hash algorithm to hash the certificate and eventually signs it with its private key. The generated developer certificate has as its only purpose to sign applications. When a developer creates an application via Apple's development tool Xcode and he/she has attached an iOS device through a USB cable, he/she is allowed to deploy the application to the iDevice [17]. Automatically after the compilation, an app container is generated containing the necessary files in order to be executed onto the iDevice. An additional file is generated with extension mobile provision. This specific file is a certificate in the form of a property list that declares the Developer ID which is the creator of the application, the Bundle ID (Unique Name) of the generated application, the target device UDID (Unique Device Identifier), the developer's public key with which the application has been signed and the permissions of the application. When a developer needs to test the application onto an iDevice he/she must first accept the developer's certificate as being legitimate through the settings of the mobile device. Using this implementation the parts of the application that have been encrypted with the developer's private key can be decrypted through the corresponding public key into the provisioned iDevice [9, 10]. The trust of this procedure is sealed with the valid certificate issued by Apple. The signed executable contains an embedded property list file, known as *entitlement*, which defines the application's Bundle ID, the Developer's ID and the permissions of the application. The entries of that file is a subset of the mobile provision's file, as explained before.

5 Attack Types

The objective of this work was to evaluate the tolerance of iOS's application code protection mechanisms. Section 5.1 demonstrates all the necessary steps to execute cracked paid applications in non jailbroken iDevices. In addition to that, we were able to extend the functionality of various applications by injecting malicious libraries as add-ons into their original bundle and deploy them also into non jailbroken iDevices. For the above purposes several 32 and 64 bit iOS devices have been used with various versions of iOS 9. Our methodology is not automated. Every step is manually driven. Automating these procedures is out of the scope of this paper.

5.1 Replication

In the experiments several legitimate paid applications, available on Apple's App Store, have been used together with several cracked application from various unofficial app stores, developed with both Objective-C and Swift programming languages. The method

for loading them onto a non jailbroken iDevice consists of the following steps (illustrated in Fig. 2):

1. User must first install any application legitimate (installed via iTunes or App Store) or cracked one (3rd party repos) into a Jailbroken iDevice.
2. After having installed a legitimate application onto a 32-bit Jailbroken iOS Device, we bypass the encryption of the application's executable by dumping the decrypted parts loaded in the virtual memory to an ARMv7-A Mach-O file. Then, we patch the decryption flag. The entire procedure has been carried out using the LLDB Debugger [20]. Application's executable decryption can also be done by automatic tools [29, 30].
3. Following the previous step, we extract the generated executable from the iDevice.
4. Then, we replace the original executable file of the app container with the cracked one. Then we modify the Bundle ID (Unique application name) of the original application listed into the info plist file inside the container of the application file, with a new name that consists of the original application's name and a random suffix. The random suffix that was utilised serves to overcome the fact that every Bundle ID is reserved and cannot be re-used. It should be stressed that the aforementioned replacement of the Bundle ID will not work for applications with iCloud or Game Center extensions.
5. Every iDevice owner is obliged to create an Apple ID account in order to have access to iTunes, App Store and iCloud. An iDevice allows a limited number of accounts per device to be created without the use of a credit card. An attacker can create as many as possible Apple accounts as he/she wants and declare them as developer accounts without paying the annual fee to activate them. As a result, the fake Apple accounts remain inactive and although they cannot be used for publishing applications to the App Store they can be used for executing application that are under development to any new iDevice. The exploited vulnerability has been based on the developers' ability to deploy their own testing applications to new iOS devices, through Xcode, without any cost but by simply using an Apple ID registered to Apple's Developer Program without enrolment. Consequently, we are able to create decoy application with the same Bundle ID as that of the modified application's (Legitimate Bundle-ID + Suffix) and bind it with the developers account. We let Xcode to automatically generate a suitable team provisioning profile in order to deploy the decoy application into a non jailbroken iDevice [12].
6. Before the user launches the decoy application for the first time, he/she must accept the developers team provisioning profile in the iDevices's Preferences.
7. At this point we are able to dump the entitlement of the generated executable and merge it with the entitlement of the original one.
8. Then the Xcode tool set [3] was employed to resign the decrypted executable with our valid developer certificate, based on the entitlement of the decoys application executable. It is clear that the bind between a valid certificate, the Developer's ID, the UDID and the Bundle ID of the application, is not enough since Apple cannot ensure that the developers actually will sign only their own legitimate applications. By resigning an application the Code Signature folder is regenerated and that allows the application to be deployed in an iDevice that has approved the developer's public

key. Finally the signed cracked application has been deployed onto a non jailbroken iDevice by cheating Xcode in the sense that the cracked application has been generated by the owner of the certificate that signs it. Due to the backward compatibility of ARM processors we were able to execute the decrypted 32 Bit armv7 executable (generated by the 32 Bit architecture of Jailbroken iPhone 5) to new iDevices with 64 and 32 Bit architectures respectively.

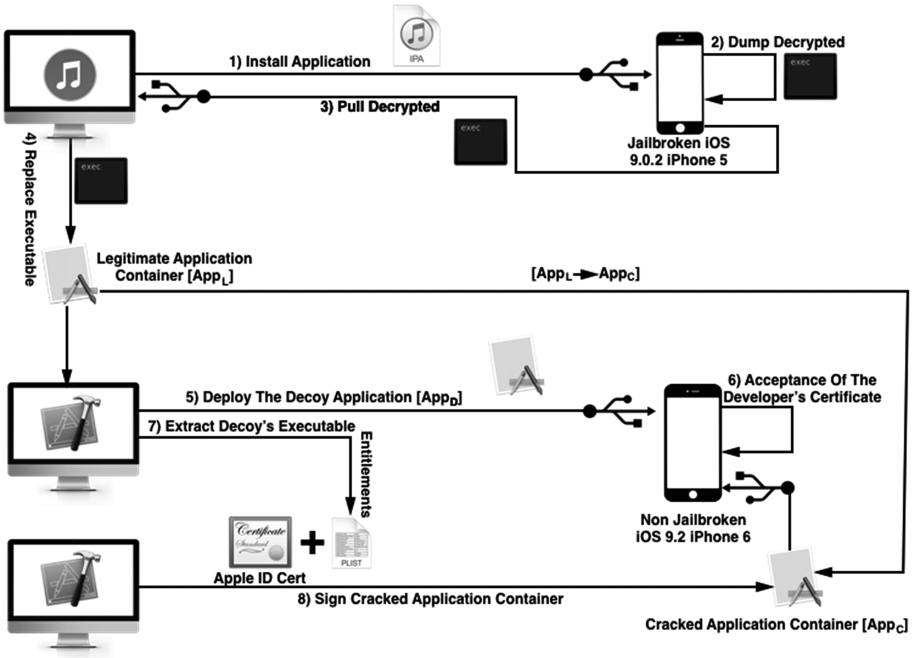


Fig. 2. Replication method

5.2 Malicious Payload Injection

Another issue that affects applications developed with the Objective-C language is the ability to hook functions of application's classes as described by Livitt [22]. After the decryption and extraction of an iOS application's executable (Step 1 Fig. 3), an attacker can reverse engineer it through static and dynamic analysis and discover the usability and functionality of its functions. Thus the attacker can take advantage of the Objective-C [13] language method calling to create dynamic libraries (Step 2 Fig. 3) and hook application's class functions and modify the passing and return values or even inject malicious payloads to them. The most suitable tool for this kind of extensions is the Theos framework [11]. This tool in combination with the iOS Software Development Kit and Cydia Substrate framework [19] is able to generate hooking dynamic libraries. For the purposes of our research we used an ARMv7-A image of Cydia Substrate suitable for both 32 and 64 Bit iOS 9 iDevices and we statically linked the Cydia Substrate to

the generated dynamic library (Step 4 Fig. 3). Then we statically linked it into the cracked executable (Step 3 Fig. 3) and place it inside the application container (Step 5 Fig. 3). Due to the additional modifications it is necessary to resign (Step 6 Fig. 3) the cracked executable and the additional dynamic libraries, with the entitlements of a decoy application as Sect. 5.1. Having the ability to hook Objective-C iOS application's class functions, an attacker can modify an application's behavior, bypass security checks, compromise application's transactions integrity and extend functionality in order to unlock premium features.

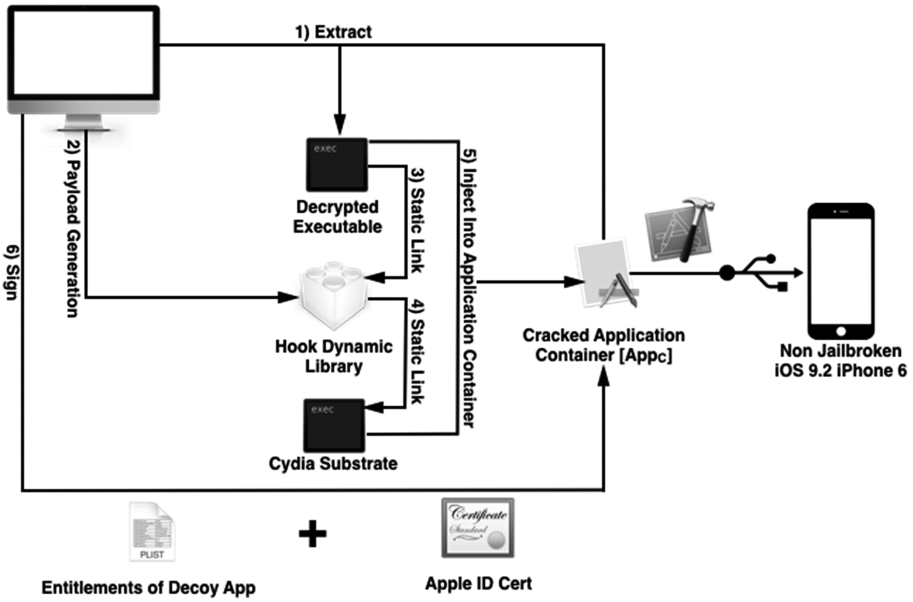


Fig. 3. Library injection

The aforementioned procedures can be performed by any owner of a non Jailbroken iOS 32 Bit or 64 bit Device with a free registration to the Apple Developer Program without enrolling his/her Apple ID and with access to a Mac or to a virtual machine of Mac OS X with Xcode installed. A further impact of signing third parties' applications as 'under test' ones, is that an attacker can unlock several developer features such as the ability to inspect memory allocations and automatically investigate and debug memory leaks throw default system tools preinstalled into any OS X.

6 Conclusions

Both cases may lead to serious financial impacts in the business model of paid and free (with in app purchases features) applications. From an economic standpoint, App Store is the largest digital distribution platform for mobile apps with the total amount of revenue from app sales since 2008 being at approximately 15 billion of United States

Dollars [7]. The use of a functional cracked application deprives the developers of the profit before taxes, which is equal to 70 % of the application's price. Also, there is a loss for Apple which amounts to the rest 30 % of the sale [28]. We were able not only to run paid and repackaged applications freely into non jailbroken iDevices but we were also able to have full access to their configuration files because we sign them as testing applications and gain paid features and bypass Apple in app purchase model by modify their data.

Running Apple ID signed applications onto not modified iDevices enlarge the attack surface of iOS platform because in combination with exploitable memory corruptions and Kernel vulnerabilities Jailbreak developers can deploy their own vulnerable apps in order to directly attack the iOS Kernel. In this paper we leverage the opportunity for unenrolled iOS developers to run freely their under developing application into their iDevices for test purposes and we prove that cracked and repackaged applications can be executed freely into every non Jailbroken devices regardless the version of the operating system.

The immediate revocation of non enrolled developer code signature certificates will only reduce the ability of iOS device owners to use cracked or malformed application to their devices, and not to eliminate that malicious activities because of the alternative equivalent methods accomplished that with enrolled developer and enterprise accounts [8]. The only way that this type of threat can be eliminated is by robust obfuscation for any generated application's executable. Another common vulnerability we faced during our research was the lack of encrypted values into applications file settings which gave us the ability to modify values related with vulnerable in app purchases implementations. Moreover it is recommended for application developers to redevelop immediately the Objective-C applications available in the App Store to their equivalent Swift editions and for Apple the design of a pure Swift framework for all the iDevices operation system. Our research is based on framework vulnerabilities and security mechanisms implemented in mobile applications. Consequently, we aim to extend our research for Android and Windows Phone applications. For Android application a malicious payload is able to be injected into a repackaged application container with a crafted C/C++ library or with Dalvik byte code injection. Similar to Android we will try to generalize those methods to Windows Phone's Applications to inject .NET assembly code into them in order to evaluate the possibility of creation prepackaged tweaked applications for this platform. Our final objective is to categorize common vulnerabilities in applications and ways they can be exploited based on the mobile platform they are implemented in order to suggest user space integrated methods for application integrity protection suitable for any mobile operating system.

Finally, the above financial and statistical data we provided are because of the seriousness of the attack and the potential losses if an escalated attack against paid or vulnerable credit based in app purchase implementations could be done.

References

1. Code Signing Guide: Code Signing Overview. Apple Inc., 23 July 2012. Web 20 June 2016
2. App Programming Guide for IOS. Inter-App Communication. Apple Inc., 16 September 2015. Web 20 June 2016
3. OS X Code Signing In Depth: Technical Note TN2206. Apple Inc., 28 July 2015. Web 08 Dec 2015
4. App Store Review Guidelines: Apple Developer. Apple Inc., 20 June 2016
5. IOS Application Security: IOS Security 2015.2, pp. 18–19. Apple Inc., 9 September 2015. Web 10 Dec 2015
6. Kwon, B.J., Mondal, J., Jang, J., Bilge, L., Dumitras, T.: The Dropper effect insights into malware distribution with downloader graph analytics. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS 2015 (2015)
7. Forbes: Forbes Magazine, 11 January 2015. http://www.forbes.com/sites/anthonykosner/2015/01/11/apple-app-store-revenue-surge-and-the-rise-of-the-freemium/. 09 Dec 2015
8. Choosing a Membership - Support. Apple Inc., 09 December 2015. https://developer.apple.com/support/compare-memberships/
9. App Distribution Guide: Exporting Your App for Testing (iOS, TvOS, WatchOS). Apple Inc., 29 May 2016. https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/TestingYouriOSApp/TestingYouriOSApp.html#//apple_ref/doc/uid/TP40012582-CH8-SW1. 20 June 2016
10. Code Signing Guide: About Code Signing. Apple Inc., 23 July 2012. https://developer.apple.com/library/mac/documentation/Security/Conceptual/CodeSigningGuide/Introduction/Introduction.html. 20 June 2016
11. Theos: Unified Cross-platform Makefile System. Github Repository, 7 February 2016. https://github.com/DHowett/theos. 20 June 2016
12. App Distribution Guide: Launching Your App on Devices. Apple Inc., 29 April 2016. http://developer.apple.com/library/mac/documentation/IDEs/Conceptual/AppDistributionGuide/LaunchingYourApponDevices/LaunchingYourApponDevices.html. 20 June 2016
13. Objective-C: Runtime Programming Guide. Messaging, Apple Inc., 19 October 2009. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ObjCRuntimeGuide/Articles/ocrtHowMessagingWorks.html. 20 June 2016
14. Zheng, M., Xue, H., Zhang, Y., Wei, T., Lui, J.C.S.: Enpublic Apps. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security - ASIA CCS 2015, pp. 463–474 (2015)
15. Wang, T., Jang, Y., Chen, Y., Chung, S., Lau, B., Lee, W.: On the Feasibility of Large-Scale Infections of IOS Devices. In: 23rd USENIX Security Symposium, pp. 79–93 (2014). Web 10 Dec 2015
16. Wang, T., Lu, K., Lu, L., Chung, S., Lee, W.: Jekyll on iOS: when benign apps become evil. In: 22nd USENIX Security Symposium, pp. 559–572 (2013)
17. App Distribution Guide: Maintaining Identifiers, Devices, and Profiles. Apple Inc., 29 April 2016. https://developer.apple.com/library/mac/documentation/IDEs/Conceptual/AppDistributionGuide/MaintainingProfiles/MaintainingProfiles.html#//apple_ref/doc/uid/TP40012582-CH30-SW26. 20 June 2016
18. In-App Purchase Programming Guide: About In-App Purchase. Apple Inc., 21 October 2005. https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/Introduction.html. 20 June 2016
19. Cydia Substrate: The Powerful Code Modification Platform behind Cydia. SaurikIT LLC (2014). http://www.cydiasubstrate.com/. 20 June 2016

20. The LLDB Debugger: LLDB Homepage. LLVM Project, 20 June 2016. http://lldb.llvm.org/. 20 June 2016
21. Levin, J.: Mac OS X and IOS Internals: To the Apple's Core. Wiley, Indianapolis (2013)
22. Livitt, C.: Rethinking & Repackaging IOS Apps: Part 2. Bishop Fox, 4 May 2015. Web 2 Dec 2015
23. Theos and Cycript for Non-jailbroken IOS Devices. Github Repository, 17 August 2015. https://github.com/BishopFox/theos-jailed. 20 June 2016
24. Apple Developer: Apple Inc. (2015). (18 Dec. 2015)
25. Passary, A.: Apple IOS 9: Here's A List of Eligible Devices. TechTimes Inc., 10 June 2015. http://www.techtimes.com/articles/59076/20150610/apple-ios-9-heres-a-list-of-eligible-devices.htm. 20 June 2016
26. IDC: Smartphone OS Market Share. IDC Research, Inc., August 2015. www.idc.com. 8 Dec 2015 http://www.idc.com/prodserv/smartphone-os-market-share.jsp
27. McCracken, H.: Who's Winning, IOS or Android? All the Numbers, All in One Place | TIME.com. Time Inc., 16 April 2013. http://techland.time.com/2013/04/16/ios-vs-android. 20 June 2016
28. From Code to Customer: Apple Developer Program. Apple Inc (2016). https://developer.apple.com/programs. 20 June 2016
29. Clutch: Fast IOS Executable Dumper. Github Repository, 15 June 2016. https://github.com/KJCracks/Clutch. 20 June 2016
30. Esser, S.: Dumped Encrypted. Github Repository, 13 February 2014. https://github.com/stefanesser/dumpdecrypted. 20 June 2016

Author Index