Information and Software Technology 100 (2018) 165–182

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Modeling Security and Privacy Requirements: a Use Case-Driven Approach

Phu X. Mai[a], Arda Goknil[*,a], Lwin Khin Shar[c], Fabrizio Pastore[a], Lionel C. Briand[a], Shaban Shaame[b]

[a] SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg
[b] EverdreamSoft (EDS), Switzerland
[c] School of Computer Science and Engineering, Nanyang Technological University, Singapore

A B S T R A C T

Context: Modern internet-based services, ranging from food-delivery to home-caring, leverage the availability of multiple programmable devices to provide handy services tailored to end-user needs. These services are delivered through an ecosystem of device-specific software components and interfaces (e.g., mobile and wearable device applications). Since they often handle private information (e.g., location and health status), their security and privacy requirements are of crucial importance. Defining and analyzing those requirements is a significant challenge due to the multiple types of software components and devices integrated into software ecosystems. Each software component presents peculiarities that often depend on the context and the devices the component interact with, and that must be considered when dealing with security and privacy requirements. *Objective:* In this paper, we propose, apply, and assess a modeling method that supports the specification of security and privacy requirements in a structured and analyzable form. Our motivation is that, in many contexts, use cases are common practice for the elicitation of functional requirements and should also be adapted for describing security requirements. *Method:* We integrate an existing approach for modeling security and privacy requirements in terms of security threats, their mitigations, and their relations to use cases in a misuse case diagram. We introduce new security-related templates, i.e., a mitigation template and a misuse case template for specifying mitigation schemes and misuse case specifications in a structured and analyzable manner. Natural language processing can then be used to automatically report inconsistencies among artifacts and between the templates and specifications. *Results:* We successfully applied our approach to an industrial healthcare project and report lessons learned and results from structured interviews with engineers. *Conclusion:* Since our approach supports the precise specification and analysis of security threats, threat scenarios and their mitigations, it also supports decision making and the analysis of compliance to standards.

1. Introduction

Modern internet-based services like home-banking [1], music-streaming [2], food-delivery [3], and personal-training [4] are delivered through multi-device software ecosystems, i.e., software systems with components and interfaces that are executed on different types of devices including Web browsers, desktop applications, mobile applications, smart-TVs, and wearable devices. Most of the multi-device software ecosystems process private end-user data collected and stored by different devices, such as credit balance reported by banking applications, locations visited by end-users, and health status tracked by personal training applications. The adoption of multi-device software ecosystems augments security and privacy risks because of the presence of multiple attack surfaces (points at which security attacks can be executed), including malware that steals consumer and corporate data from smartphones [5] and Web applications that unintentionally expose confidential data [6]. Therefore, security and privacy have become a crucial concern in the development of software ecosystems, starting from requirements analysis to testing.

To identify the security requirements of a multi-device software ecosystem, it is necessary to take into consideration the characteristics of the specific service being developed and of the device types on which the service is going to be deployed. An example requirement of a home-banking smartphone service is that the user should automatically log off when the phone screen is locked to prevent phone thieves from accessing the bank account. This requirement is inappropriate for other types of services, e.g., personal training services which are used by runners and thus should be accessible without logging in, even after a screen

* Corresponding author.
E-mail addresses: xuanphu.mai@uni.lu (P.X. Mai), arda.goknil@uni.lu, goknil@svv.lu (A. Goknil), lkshar@ntu.edu.sg (L.K. Shar), fabrizio.pastore@uni.lu (F. Pastore), lionel.briand@uni.lu (L.C. Briand), shaban@everdreamsoft.com (S. Shaame).

https://doi.org/10.1016/j.infsof.2018.04.007
Received 9 November 2017; Received in revised form 19 February 2018; Accepted 16 April 2018
Available online 19 April 2018
0950-5849/ © 2018 Elsevier B.V. All rights reserved.

lock (which normally happens while running). Examples of device specific characteristics that impact on security requirements include Web applications running on dedicated servers that are always online and thus prone to brute force attacks via the network. Mobile applications, instead, are often idle or offline, but they usually run on a device that is potentially shared with malicious applications inadvertently installed by end-users. Such applications can steal private data if it is not properly protected (e.g., through encryption). Therefore, it is crucial to precisely model and analyze security and privacy requirements of such multi-device software ecosystems early in their development.

In this paper, we propose, apply, and assess a use case-driven modeling method that supports the specification of security and privacy requirements of multi-device software ecosystems in a structured and analyzable form. Use cases are one of the most common means adopted by software engineers to elicit requirements because they ease the communication between stakeholders [7]. Therefore, to achieve widespread applicability, the need for integrating security requirements with use case modeling warrants the development of a use case-driven, security requirements modeling method that is, in our context, tailored to the development of multi-device software ecosystems.

Considerable research has been devoted to eliciting and analyzing security requirements using various forms of use cases (e.g., abuse cases [8,9], security use cases [10], and misuse cases [11–14]). However, the applicability of these approaches in the context of security and privacy requirements modeling for multi-device software ecosystems shows limitations with respect to (1) their support for explicitly specifying various types of security threats (a security threat is a possible event that exploits a vulnerability of the system to cause harm), (2) the definition of threat scenarios (a threat scenario is a flow of events containing interactions between a malicious actor and system to cause harm), and (3) the specification of mitigations for these threats.

These three features are essential in the type of business context we target where it is required to explicitly identify the threat scenarios that may affect important business operations in order to identify appropriate mitigation schemes and trade-offs between functional requirements and security and privacy concerns. It is also expected that such security requirements, specified in a structured and analyzable form, provide support for security testing, for example by helping with the identification of attack surfaces. In addition to specifying security threats, a common practice in many environments requires mitigation schemes to be documented for the stakeholders to demonstrate compliance with applicable security and privacy standards and regulations. However, existing approaches lack reusable templates to specify such mitigation schemes.

The goal of this paper is to address the above challenges by proposing a use case-driven, security requirements modeling method called *Restricted Misuse Case Modeling (RMCM)*, which adapts existing methods and extends them. In RMCM, we employ misuse case diagrams proposed by Sindre and Opdahl [13] to model security and privacy requirements in terms of use cases. Misuse cases describe attacks that may compromise use cases; security use cases specify how to mitigate such attacks. For eliciting security threats and threat scenarios in a structured and analyzable form, we adopt the Restricted Use Case Modeling method (RUCM) proposed in [15] to write use case specifications. RUCM is based on a template and restriction rules, reducing ambiguities and incompleteness in use cases. It was previously evaluated through controlled experiments and has shown to be usable and beneficial with respect to making use cases less ambiguous and more amenable to precise analysis and design [16–23]. However, since RUCM was not originally designed for modeling security and privacy requirements, we extend the RUCM template with new restriction rules and constructs, targeting the precise modeling of security threats. Further, we provide a template for mitigation schemes and three mitigation schemes that are pre-specified with standard and secure coding methods for mitigating common security threats. They can be readily used and revised as necessary.

In this paper, we focus on one important aspect of privacy: the security of personal data. More specifically, we support the modeling of requirements regarding three data protection goals [24]: confidentiality, integrity and availability. The definition of methods to model other data protection requirements (e.g., unlinkability, transparency, and intervenability) and to target privacy-related activities other than information processing (e.g., dissemination or collection [25]) is out of the scope of this paper (related work includes reports and regulations on data minimization, collection limitation, and purpose specification [26–29]).

Leveraging on the analyzable form of our models, RMCM employs Natural Language Processing (NLP) to report inconsistencies between a misuse case diagram and its RMCM specifications, and to analyze the compliance of such specifications against the provided RMCM templates. NLP is also used to identify and highlight the control flow leading to different threat scenarios and the steps in RMCM specifications that refer to interactions between malicious actors and the system. The latter provides security testers with information about attack surfaces on which security testing should focus. To summarize, the contributions of this paper are:

- RMCM, a security requirements modeling method supporting the precise and analyzable specification of security threats, threat scenarios, and their mitigations, in the context of use case driven development of multi-device software ecosystems;
- a practical toolchain, available at our tool website [30], including (1) a component that extends Papyrus [31] to support misuse case diagrams, (2) a component that extends IBM Doors [32] to support misuse case specifications and mitigation schemes in the RMCM templates, and (3) a component relying on NLP to detect inconsistencies among these artifacts;
- a case study demonstrating the applicability of RMCM in a realistic development context involving multiple service and software providers in the healthcare domain.

This paper is structured as follows. Section 2 introduces the context of our case study to provide the motivations behind RMCM. Section 3 describes the limitations of state-of-the-art approaches that we identified by concretely applying these approaches on our industrial case study. Section 4 discusses the related work. Section 5 provides an overview of RMCM. Section 6 focuses on the use case extensions in RMCM. In Section 7, we present our tool support. Section 8 reports on our industrial case study, from which we draw conclusions on the benefits and applicability of the proposed approach. Section 9 concludes the paper.

2. Context and motivation

The work presented in this paper is part of a European Union (EU) project, i.e., EDLAH2 [33], in the healthcare domain. The project brings academic institutions and software development companies together in a consortium to enhance the lifestyle of elderly people through a *gamification-based* approach. Gamification transforms activities that we are normally reluctant to do, e.g., exercising regularly, into a competition [34]. The objective of the EDLAH2 project is to provide a set of gamification-based services on mobile devices that engage and challenge clients (elderly people) to improve their physical, mental, and social activities.

To achieve this objective, the EDLAH2 consortium is developing a multi-device software ecosystem, i.e., a set of software components that can run on multiple types of systems and devices, which include mobile and wearable device applications (services). In EDLAH2, the mobile applications are used to incentivize elderly people to perform intellectual activities (e.g., solving logic-based games including Sudoku), while the wearable device applications are used to track physical activities (e.g., tracking heartbeat and footsteps via a bracelet device). These applications collect data to store in a central data repository in a server. The EDLAH2 website, i.e., *iCare* [35], provides access to the data collected by the applications. It allows the carers of the clients to create user accounts and to configure the system (e.g., selecting mobile applications to install on the client's devices). The EDLAH2 system has thus a multi-tier and multi-device architecture, in which mobile applications interact with Web applications (i.e., software applications running on a Web server), Web applications interact with databases and third-party software, and mobile applications interact with mobile device data storage (e.g., SQLite or SD cards). In this paper, we use the multi-device software ecosystem developed for EDLAH2 as a case study to motivate, illustrate, and assess RMCM.

The objective of EDLAH2 entails that end-users (i.e., elderly people) provide access not only to their personal data but also to their daily activities. Hence, EDLAH2 is representative of contexts where engineers face the significant challenge of defining and ensuring security and privacy requirements in systems that process users' private data which is produced and shared by multiple components. Ensuring security and privacy in such contexts is complicated by multiple factors such as the presence of multiple components, communication over networks, and complex information flows involving multiple actors (e.g., end-users, mobile apps, third party software, and Web apps).

Multi-device software ecosystems like EDLAH2 are thus exposed to numerous security threats such as information disclosure, information modification, unauthorized access, and denial of service. A security threat becomes a reality in the presence of vulnerabilities that can be exploited by an attacker. Vulnerabilities might be introduced because of different reasons, ranging from the incomplete identification of security requirements to the adoption of bad programming practices, or the misconfiguration of software components and libraries. For example, disclosure of customer information may depend on improper requirements analysis, (e.g., the software analyst does not realize that the system should not expose the e-mail addresses registered on the platform). SQL injection (SQLI) attacks often depend on bad programming practices (e.g., SQL statements created without relying on standard libraries that include sanitization mechanisms). Cross site scripting (XSS) vulnerabilities may depend on misconfiguration of default Javascript protection options. To systematically determine countermeasures which mitigate security threats, it is thus crucial to explicitly model both the activities that the system should perform to protect itself and the potential security threats that depend on the type of software components being developed.

The current development practice in EDLAH2 is use case-driven and involves UML use case diagrams and use case specifications for describing functional requirements. Fig. 1 depicts part of the use case diagram of EDLAH2. *Client* and *Carer* are the main actors of the system while *Bracelet, Game App, Browser* and *Skype* are the secondary actors representing the third-party apps. The use cases describe seven main functionalities: *get fitter, play games, do social activities, get rewards, log in, create account*, and *configure system*.

A use case specification contains a detailed description of a use case and usually conforms to a template [36–38]. The Cockburn template [36] had been followed so far to document EDLAH2 use case specifications in EDLAH2. Fig. 2 shows two examples of such specifications that are part of EDLAH2. *Log in* describes how the carer logs into the system via the iCare website. *Get Fitter* describes how the client checks his physical activities and condition (e.g., heart beat rate, number of steps and minutes of walking) as measured by the wearable device (i.e., bracelet).

The sample use case specifications reflect scenarios in which the client and the carer use the system as expected. From these scenarios, one may also observe that the system accesses, processes, propagates, and stores the user's private information. As a result, security and privacy concerns need to be elicited such that stakeholders can take appropriate actions where needed.

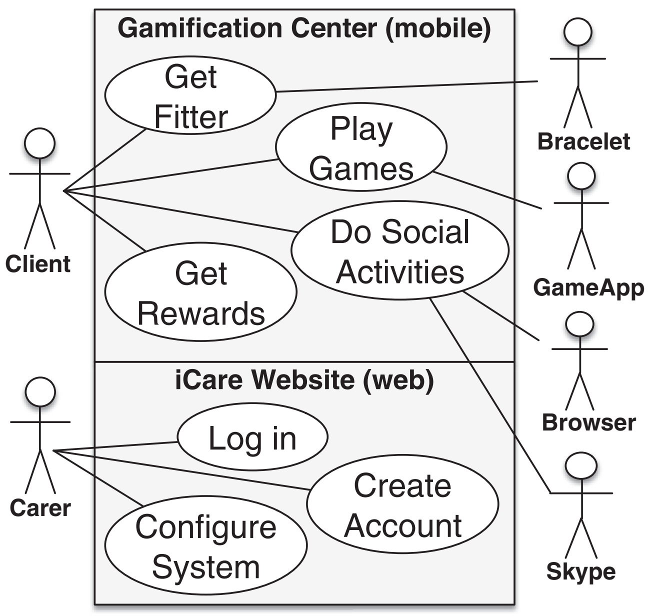


Fig. 1. Part of the use case diagram of EDLAH2.

```
USE CASE Log in
Precondition The system displays the login screen.
Basic Path
1. The carer enters the user name and password in the login form.
2. The system checks in the browser if the user name and password are valid.
3. The system builds a database query using the user name and password.
4. The system evaluates the query in the database.
5. The system checks that the query is successful.
6. The system displays the welcome message.
Postcondition The carer has successfully logged in the system.
Alternative Paths
2a. The entered user name or password is invalid.
   2a1. The system displays the wrong user name or password message.
5a. The query is unsuccessful.
   5a1. The system displays the database error message.

USE CASE Get Fitter
Precondition The client has been successfully logged into the system.
Basic Path
1. The client requests to get current measurement.
2. The system receives the heart beat rate data from the bracelet.
3. The system checks whether the received heart beat rate data is correct.
4. The system stores the received data.
5. The system sets one point as a reward for the client.
6. The system displays the received heart beat rate data.
7. The system displays one point as a reward.
Postcondition The heart beat rate data and reward have been stored.
Alternative Paths
1a. The client requests to get activity data.
   1a1. The system receives the client's activity data from the bracelet.
   1a2. The system checks whether the activity data is correct.
   1a3. The system stores the activity data.
   1a4. The system sets two point as a reward for the client.
   1a5. The system displays the activity data.
   1a6. The system displays two points as a reward.
    1a3a. The system displays the error for incorrect activity data.
4a. The system displays the error for incorrect heart beat rate measurement.
```

Fig. 2. Sample use case specifications for part of the EDLAH2 system.

3. Definition of security and privacy requirements of a software service ecosystem: Practical challenges

Standard use case templates, such as Cockburn's, are insufficient to document security and privacy concerns in use case specifications [13,14]. One state-of-the-art approach for eliciting security concerns, together with functional requirements, provides a misuse case specification template [11,13] which extends a use case template with additional notions such as *misuse* and *mitigation point*. We applied this template in the context of the EDLAH2 project and attempted to elicit

```
1  MISUSE CASE Get Unauthorized Access
2  Precondition At least one client account has already been created in the
   system.
3  Basic Path
4  1. The crook tampers with the values in the login URL.
5  2. The crook submits the tampered URL directly to the system.
6  3. The system builds a query using the values provided in login URL.
7  4. The system evaluates the query in the database.
8  5. The system checks that the query is successful.
9  6. The system displays the welcome message.
10 Postcondition The crook has gained some privileges.
11 Alternative Paths
12 5a. The query is unsuccessful.
13   5a1. The system displays the database error message, revealing some
          information about the database structure.
14    5a2. The crook tampers with the values in the login URL again
          based on the exposed information.
15    5a3. The crook submits the tampered URL directly to the system
          until the system checks that the query is successful.
16 5a3. The crook reaches maximum number of login attempts.
17    5a3a. The system displays the error message for login.
18 Mitigation Points
19 mp1. In Step 3, the system sanitizes the values before building the query.
20 mp2. In Step 5a1, the system does not replay the exact database error message
        and instead, it displays only non-confidential information.
21
22 MISUSE CASE Expose Information from Mobile
23 Precondition The mobile device also has a malware installed.
24 Basic Path
25 1. The malware requests access to user data stored in the system.
26 2. The system accepts the request.
27 3. The system sends user data to the malware.
28 Postcondition The malware has obtained user's private information.
29 Alternative Paths
30 2a. The system rejects the request.
```

Fig. 3. Sample misuse case specifications for part of the EDLAH2 system.

some of the security and privacy concerns for the use case specifications reported in Fig. 2. Some results of our attempt are shown in Fig. 3. The *Get Unauthorized Access* and *Expose Information from Mobile* misuse case specifications we targeted are similar to the example given in [13]. The basic and alternative paths in Fig. 3 describe the sequence of actions that malicious actors go through to cause harm. The mitigation points document the actions in a path where the misuse case can be mitigated (Lines 19–20 in Fig. 3).

Based on this attempt, we identified three challenges that need to be considered when capturing security and privacy requirements in use case-driven development of multi-device software ecosystems:

Eliciting security threats in an explicit, precise form (*Challenge 1*). We noticed that although the template we applied supports specifying various threats, it does not support their specification in a precise and unambiguous manner. This is the same for other related approaches such as [8,10,11,13]. We identified three main limitations, reported in the following.

First, existing templates do not provide glossary or keywords for specifying common security threats. For instance, the *Get Unauthorized Access* misuse case in Fig. 3 corresponds to unauthorized access via SQLI (for categorizing the security threats, we follow the common, well-known terminology given in OWASP [39,40]). In the specification, the term "SQL" was not even used. Likewise, the *Expose Information from Mobile* misuse case corresponds to information disclosure due to insecure data storage.

Second, existing templates do not provide a precise and systematic way to determine malicious actors in the specification. However, providing security extensions or keywords for precisely specifying common security threats, specific to the device type, would be useful. It would facilitate unambiguous communication among stakeholders and support various automated analyses, including security testing (e.g., identify the attacks that might hit a specific functionality and verify if its implementation properly prevents the attacks by simulating the attacker behaviour).

Third, existing templates do not explicitly distinguish between malicious actor-system interactions and other types of interactions. For instance, the steps in Lines 4–5 in Fig. 3 correspond to the malicious actor-system interactions, whereas the steps in Lines 6–9 correspond to the system's internal state changes. The interactions between malicious actors and the system contain information about the attack surfaces. But since the specification provided in Fig. 3 does not make this important difference, it might be difficult for a security tester to precisely determine where the attack surface is. In this case, the attack surface consists of the parameters in the login URL.

Eliciting threat scenarios in a structured and analyzable form (*Challenge 2*). The existing templates have two shortcomings in eliciting threat scenarios. First, they do not have any explicit control flow structure. For instance, the *Get Unauthorized Access* misuse case in Fig. 3 tries a list of user name and password tuples iteratively until the malicious user logs into the system to get privileges. Since we do not have any explicit loop structure in the template we use for misuse cases in Fig. 3, we tried to describe the loop condition for the threat in non-restrictive natural language ('... until the system checks that...' in Line 15 in Fig. 3). However, it is not clear where the iteration starts in the execution flow. Second, this does not allow to discern different types of scenarios — scenarios that a malicious actor may follow to successfully harm the system and scenarios that may not result in such harm. For instance, in the *Get Unauthorized Access* misuse case in Fig. 3, there are two alternative paths — the one starting from Line 12 leads to the scenario where the malicious actor harms the system and the other one starting from Line 16 leads to the scenario where the malicious actor fails to harm the system. As a result, it may not be easy for the stakeholders or an analysis tool to distinguish control flows and conditions leading to threat scenarios. Therefore, such specifications can be ambiguous and cannot support automated analyses.

Eliciting mitigation schemes (*Challenge 3*). After identifying security threats in threat scenarios, it is crucial to specify mitigation schemes matching these threats to demonstrate that the software design complies with applicable security and privacy standards and regulations. Such mitigation schemes provide the developers with guidance on how to prevent security threats specified in misuse cases. Different security threats and threat scenarios often share common mitigation methods and guidance. For instance, the two different security threats — information disclosure via SQLI and unauthorized access via SQLI — can both be mitigated by parameterizing the SQL queries. Existing work only supports specifying the flow of events mitigating each specific threat scenario (see Section 4). Such flows of events are embedded in misuse case specifications where one should specify the mitigation points (Lines 18–20 in Fig. 3). There is no structured way to specify the guidance for developers to mitigate security threats. In other words, there is a lack of template support for specifying mitigation schemes that can be reused and adapted for various security threats.

In this work, we focus on how to best address these three challenges in a practical manner. Automated test generation for security testing is one potential application of our method, but we leave it out for future work.

4. Related work

There are numerous approaches in the literature to model security and privacy requirements [41–46] [47,48]. In a comprehensive literature review [42], security requirements engineering methods were distinguished across six categories: *multilateral* (e.g., [49–51]), *UML-based* (e.g., [13,52,53]), *goal-oriented* (e.g., [54–60]), *problem frame-based* (e.g., [61–66]), *risk/threat analysis-based* (e.g., [67–72]), and *common criteria-based approaches* (e.g., [73–75]). The multilateral approaches follow the principles of multilateral security [76] and focus on consolidating and reconciling the views of multiple stakeholders on the security and privacy requirements. Goal-oriented methods guide engineers towards the refinement of security [54–59] and privacy [54,60] features from high level requirements and have been applied to check whether a system meets its security requirements [54,58], to identify trade-offs between security and other requirements [59], and to identify the architecture that suits the given goals [60]. The risk/threat analysis-based approaches consist of the identification of security goals and threats and focus on the analysis of the effects of these threats based on system specific information like trust relations among parties [68], likelihood of incidents [67], or data storage locations [77].

The problem frame-based approaches make use of the ideas underlying Jackson's problem frames [78], and have been adopted to model both security [61–65] and privacy requirements [66]. The common criteria-based approaches follow an international standard for information technology security evaluation [73].

Since our modeling method is based on misuse cases [13] extending UML use case diagrams, it can be considered a *UML-based approach*. It also relies on a form of *risk/threat analysis* to capture various security threats, to elicit threat scenarios in a structured form, and to specify mitigation schemes. In the following, we discuss the previous works in the literature with respect to the challenges identified in Section 3, mainly focusing on the UML-based and risk/threat analysis approaches.

Eliciting security threats in an explicit, precise form. Most of the existing approaches rely on UML diagrams and use case specification templates to capture security threats. McDermott and Fox [8,9] propose *abuse cases* to describe harmful interactions (i.e., security threats) between a system and malicious actors, but relations between abuse cases and other types of requirements are not described. Sindre and Opdahl [13] extend UML use case diagram with *misuse cases* and *security use cases* to model security threats (i.e., *misuse*), security-related requirements (i.e., *threat mitigation*) and other functional requirements. Alexander discusses automation to support misuse case diagrams [79,80] and reports experiences with misuse case diagrams in an industrial setting [81]. Rosado et al. [82] show how misuse case diagrams are employed to model the security requirements of a grid application, while Rostad [12] extends misuse case diagrams with the notion of *vulnerability*, i.e., a weakness that may be exploited by attackers. Misuse case diagrams can be employed to represent misuse cases, security use cases and their relations, but not to capture security threats in misuse cases. To address this problem, Sindre and Opdahl [14] adapt a use case specification template for detailed textual descriptions of threat scenarios. Common criteria-based approaches [74,75] apply the adapted template with misuse case diagrams [13,14] to elicit security threats and requirements. This template is extended for misuse case generalization [83] and reuse [84]. Deng et al., [85] employ the misuse case template [14] in their privacy threat analysis framework to elicit privacy threat scenarios. Omoronyia et al., [86] introduce two new fields into the template to highlight contextual properties of privacy misuse cases. Firesmith [10] proposes a similar template for *security use cases* which represent security-related requirements combined with a form of threat scenarios. However, these templates do not provide any construct or restriction rule to capture security threats in a precise and analyzable form to support automated analyses. El-Attar [87,88] proposes an approach to guide the analysts towards developing consistent misuse case diagrams and specifications. El-Attar's specifications can be automatically processed thanks to the presence of keywords, e.g., *misuse case* and *include*, that are used within the fields present in common use case templates. However, the keywords introduced by El-Attar do not support capturing security threats in an explicit form.

Goal-based approaches, such as KAOS [89], Secure Tropos [58], and attack trees [90], provide templates for specifying threats and common security goals. For instance, KAOS [89] models threats as fault trees, also referred to as obstacle trees, in which the root is a negation (anti-goal) of a security goal. Based on the anti-goals, the approach determines how attackers may harm the system under design. PriS [60] is a goal-oriented methodology that focusses on privacy requirements. It includes a set of patterns that describe the processes to put in place in order to achieve a specific privacy goal, and provides a method to determine the architectural solution that fits the requirements. PriS mostly focusses on goals, and provides little support to model privacy threats. CORAS [67], a risk analysis-based approach, rather focuses on the risks posed by security threats and on guiding the analysts in performing security risk analysis. It supports eliciting security threats using UML-like diagrams and some security-related notations at a high level. Rashid et al. [91] employ the grounded theory method [92] and incident fault trees [93] to discover and document emergent security threats which are implicit within and across a variety of security incidents. Risk-analysis approaches targeting privacy concerns instead focus on the definition of questionnaires that support analysts in the identification of privacy risks, but do not provide templates or models to capture the risks in a structured form [77,94].

Other approaches focus on security policy violations [52,95,96] and conflicting security objectives [49,51]. SecureUML [52] is a modeling language for the model-driven development of secure, distributed systems based on UML, but its modeling support is limited to specifying policies for access control threats. Breaux et al., [95,96] propose a methodology which maps privacy requirements in natural language text to a formal language in description logic to detect conflicting privacy requirements causing threats. The analysis of conflicting privacy requirements is limited to the detection of conflicts between which data is actually processed and which data processing activity is declared in the policy. Multilateral approaches [49,51] analyze general security and privacy needs of all the stakeholders of a system-to-be to consolidate different stakeholders' views on security threats. These approaches focus on identifying and resolving conflicts between different security goals of stakeholders. Their artifacts (e.g., UML models and attack trees) may explicitly capture interactions among security requirements as well as between security and functional requirements. Automated analysis of the artifacts to identify conflicts and ambiguities is also possible with some form of formalization work. However, the existing multilateral approaches provide rather conceptual frameworks. For example, SQUARE [51] only recommends to apply existing techniques that best suit the project at hand to model various threats.

In general, all the above-mentioned approaches provide systematic methods for modeling security threats and requirements in terms of security goals, anti-goals, attack trees, and/or access control policies at a high level. However, they do not focus on capturing security threats in a precise, detailed, and structured form. Therefore, automated analyses of the artifacts produced by these approaches are often not possible without incorporating additional formal notations such as temporal logics or logical formulas, which can be a tedious requirement even for developers and security analysts. In contrast to these approaches, our approach focuses on providing well-defined templates, restriction rules, and keywords to precisely capture various security threats in misuse case specifications. As our approach only requires knowledge on use case specification methods, it is relatively easy for stakeholders to participate and communicate. The templates, restriction rules and keywords are proposed such that they enable automated analyses (to identify conflicts and ambiguities) on the generated artifacts (i.e., misuse case diagrams and specifications).

Eliciting threat scenarios in a structured and analyzable form. The templates proposed for misuse cases [14,87,88], security use cases [10] and abuse cases [9] extend the common use case templates in the literature to elicit security requirements. But in general they do not provide any extension or any control flow structure to systematically identify and capture various threat scenarios. Some approaches employ UML models to use the control flow structures of UML in eliciting threat scenarios. For instance, Whittle et al. [97] propose the use of sequence diagrams for the analysis of threat scenarios in misuse cases, while Sindre et al. [98] incorporate malicious activities and malicious actors in UML activity diagrams to model potential attacks. Secure Tropos [58] incorporates security rules in UML sequence diagrams. Song et al., [99] propose to use aspect sequence diagrams to model access control-related security requirements. CORAS [67] employs UML sequence and activity diagrams to model the behavior of the system under attack. UMLsec [100] combines several UML diagrams (e.g., statecharts and interaction diagrams) for modeling and analyzing threat scenarios. It also proposes some UML extensions (i.e., stereotypes, constraints, tagged values defined in a UML profile) to capture security concepts.

Attack trees [101] and Microsoft's threat modeling approach [90], which can be considered as a special case of goal-based approaches, represent attacks or security threats against a system in a tree structure. The root node in the tree structure represents the attacker's goal while leaf nodes represent different ways of achieving that goal. A tree reflects the sequences of actions that must be carried out (a threat scenario) to realize a threat. However, it does not capture the conditions that must be met for the attack to happen. Problem frames are also used to model security threats [61,62,102]. They are basically diagrams representing the assets to be protected, the malicious subjects, the potential vulnerabilities of the system that the malicious subjects may exploit, and the environment through which the malicious users interact with the assets. Hatebur et al., [62] equip problem frames, representing threat scenarios, with preconditions and postconditions. Preconditions are specified using logical formulas to express what conditions must be met for a frame to be applicable.

Although the above-mentioned approaches provide diagrams or formalizations that employ a form of control flow structures in describing threat scenarios in a structured form, they do not provide any systematic way to distinguish scenarios that cause harm from those that do not. In addition, these approaches generally aim to aid the analysts for modeling threat scenarios using sequence/activity diagrams or formal methods. It is not clear whether stakeholders, who are not developers and with different levels of technical competence, can use these approaches. Since our approach is based on use cases, it allows different types of stakeholders to participate and communicate. We extend the RUCM template because it already provides control flow structures, e.g., *'do...while'* and *'if...then...else...'*, which can also be used for modeling threat scenarios. The new extensions we proposed capture success and failure scenarios in a structured and analyzable form.

Eliciting mitigation schemes. The template proposed by Sindre and Opdahl [14] supports mitigation points where one can specify the flow of events mitigating each specific threat scenario. There is, however, no structured way to specify mitigation schemes, i.e., the guidance and methods for developers to mitigate security threats. Our approach is the first that provides template support for specifying mitigation schemes, which can be reused for mitigating various security threats. Further, to the best of our knowledge, the current approaches do not provide a way for stakeholders to demonstrate compliance of their software design against applicable security standards, which is an important requirement in many contexts. Using our mitigation and security use case templates, this compliance-traceability requirement can be met.

5. Overview of our modeling method

The process in Fig. 4 presents an overview of our modeling method, RMCM. It is designed to address the challenges stated above in the use case-driven development context we described for multi-device software ecosystems, and builds upon and integrates existing work. The RMCM output is a misuse case diagram, use case specifications, security use case specifications, misuse case specifications, and mitigation schemes.

In Step 1, *Elicit requirements as use cases, security use cases and misuse cases*, the analyst elicits functional and security requirements relying on a misuse case diagram and the extended RUCM template (hereafter RMCM template), which are detailed in Section 6. Functional requirements and security requirements are captured in the misuse case diagram while it is further detailed in use case, security use case and misuse case specifications (*Challenges 1 and 2*). While use cases capture functional requirements, security use cases capture security

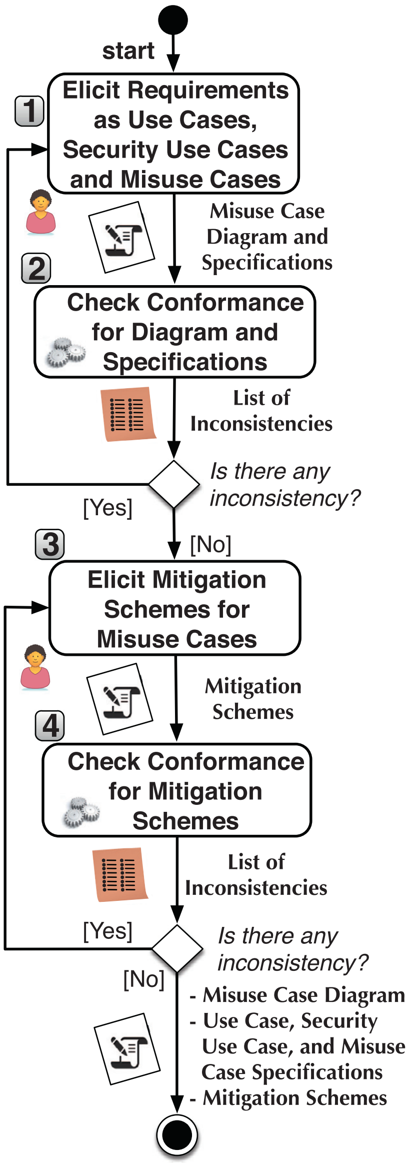

start
1
Elicit Requirements as Use Cases, Security Use Cases and Misuse Cases
Misuse Case Diagram and Specifications
2
Check Conformance for Diagram and Specifications
List of Inconsistencies
Is there any inconsistency?
[Yes]
[No]
3
Elicit Mitigation Schemes for Misuse Cases
Mitigation Schemes
4
Check Conformance for Mitigation Schemes
List of Inconsistencies
[Yes]
Is there any inconsistency?
[No]
- Misuse Case Diagram
- Use Case, Security Use Case, and Misuse Case Specifications
- Mitigation Schemes

Fig. 4. Approach overview.

countermeasures addressing potential attacks, which are themselves represented with misuse cases.

In Step 2, *Check conformance for diagram and specifications*, RMCM-V (Restricted Misuse Case Modeling - Verifier), the tool we developed for RMCM, automatically checks the consistency between the misuse case diagram and specifications, and between the specifications and the RMCM template. It relies on NLP. If there is any inconsistency, the analyst updates the diagram or specifications (Step 1). Steps 1 and 2 are iterative: the specifications and diagram are updated until the specifications conform to the RMCM template and they are consistent with the diagram.

In Step 3, *Elicit mitigation schemes for misuse cases*, mitigation schemes are elicited for the security threats specified in misuse cases (*Challenge 3*). Different from security use cases specifying the flow of events mitigating a specific threat scenario, mitigation schemes provide the secure coding methods adopted by the system, guidelines on how to educate users and other mechanisms to prevent various security threats in general. In Step 4, *Check conformance for mitigation schemes*, RMCM-V automatically checks whether the mitigation schemes conform to the mitigation template. Steps 3 and 4 are also iterative: the mitigation schemes are updated until they conform to the template.

The proposed method enables engineers to capture security threats and countermeasures. Risk analysis, i.e., ranking and prioritizing of security threats, is out of our scope. However, in contexts where risk analysis drives the engineering process (e.g., to prioritize test cases [103,104]), the proposed approach can be integrated with existing risk analysis techniques, for example, techniques that rely on misuse case diagrams [105]. Similarly, the Common Vulnerability Scoring System (CVSS) [106] can be used to evaluate the risks due to vulnerabilities affecting the deployed production system.

In the following sections, we provide a detailed description of the steps of the proposed approach.

6. Capturing security requirements

In this section, we describe the artifacts produced by RMCM. We discuss how they were extended, compared to what was proposed in existing work, and illustrate how they address our three challenges with the running example.

6.1. Use case diagram with misuse case extensions

To capture misuse cases, security use cases, use cases, and their relationships, RMCM relies on the misuse case extensions proposed by Sindre and Opdahl [13] for use case diagrams. We made this choice for RMCM because of the explicit representation of misuse cases, security use cases, and their relationships (i.e., *threaten* and *mitigate*). In the following, we briefly introduce our extensions. The reader is referred to [13] for further details. Fig. 5 depicts part of the misuse case diagram for EDLAH2.

As shown in Fig. 5, misuse cases, i.e., sequence of actions that a malicious actor can perform to cause harm, are greyed to distinguish them from use cases. Likewise, malicious actors (e.g., Malicious app) are distinguished from benign actors (e.g., Carer) and labeled with the keyword ‘malicious’. The UML stereotype ≪security≫ is used to distinguish security use cases that are countermeasures against misuse cases. In addition to the use case relationships (e.g., *include* and *extend*), *mitigate* is used for specifying the relationships between security use cases and misuse cases; and *threaten* is used for specifying the relationships between misuse cases and use cases [13]. For instance, in Fig. 5, *Validate Website Inputs* mitigates *Get Unauthorized Access via SQLI*, which threatens *Log in*. *Expose Information via Insecure Data Storage* is an abstract misuse case that is extended by some concrete misuse cases threatening *Get Fitter, Play Games, Do Social Activities*, and *Get Rewards*. For space reasons, we do not show them here. Please refer to our website [30] for details.

6.2. Misuse case and security use case specifications

Regarding misuse case specifications, to elicit security threats in a precise form and to elicit threat scenarios in a structured and analyzable form (*Challenges 1 and 2*), we propose the RMCM template, an extension of the RUCM template, shown in Table 1, and new restriction rules, shown in Table 2. The misuse case specifications are elicited using this template, further using the new restriction rules in addition to the original ones. These template and restriction rules are designed to make (mis)use case specifications explicit, precise, and analyzable by restricting the use of natural language and by using specific keywords. Our extensions specifically target the modeling of security and privacy concerns for multi-device software ecosystems.

Fig. 6 shows two simplified misuse case specifications of EDLAH2 written in RMCM, with all the RMCM keywords written in capital letters.

The original RUCM template provides basic and alternative flows which we adapted as *Basic Threat Flow, Specific/Bounded/Global Alternative Flow* and *Specific/Bounded/Global Alternative Threat Flow* (see Table 1). Threat flows specify unwanted incidents. Different from a basic flow in a use case specification, which describes a nominal scenario for an actor to use the system as intended, a basic threat flow

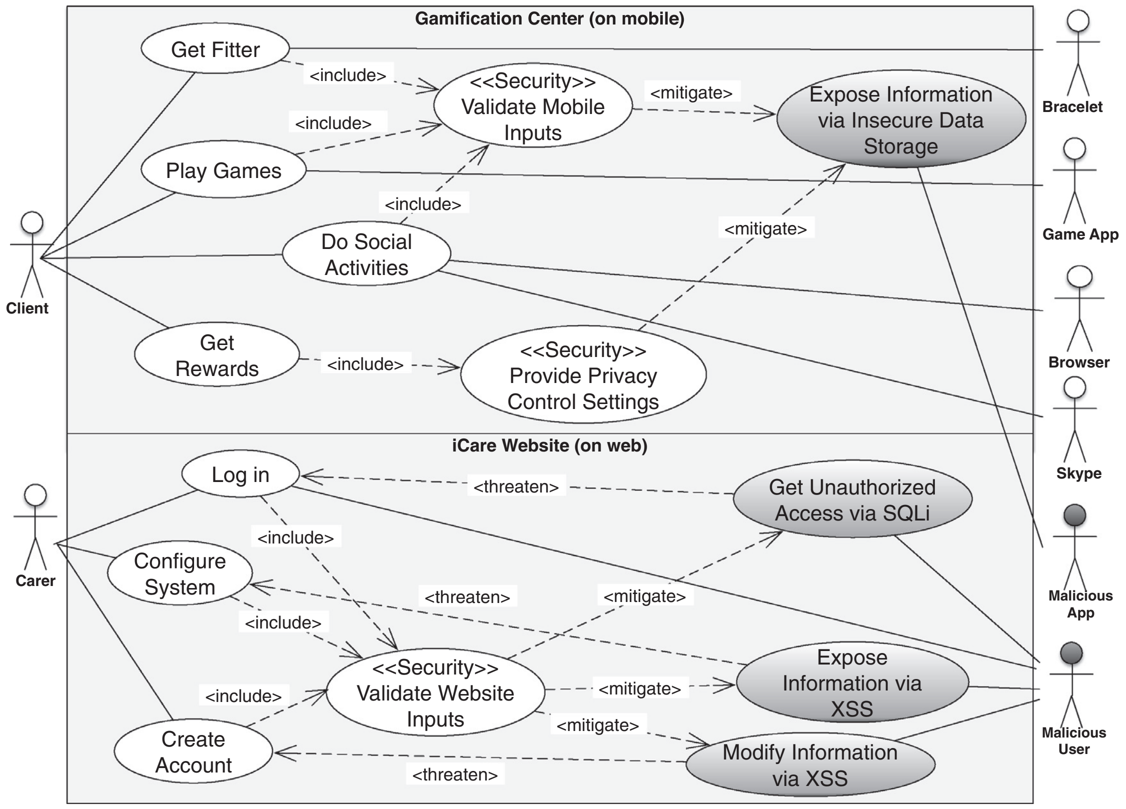


Fig. 5. Part of the misuse use case diagram for EDLAH2.

Table 1
Restricted misuse case modeling (RMCM) template.

Misuse Case Name	The name of the misuse case.	
Brief Description	Summarizes the misuse case in a short paragraph.	
Precondition	What should be true before the misuse case is executed.	
Primary Actor	The actor which initiates the misuse case.	
Secondary Actors	The actors which interact with the system to accomplish the misuse case.	
Dependency	Include and extend relationships to other (mis)use cases.	
Generalization	Generalization relationships to other misuse cases.	
Threats	Threaten relationships to use cases.	
Assets	The assets (potentially) impacted by this threat.	
Basic Threat Flow	Specifies the main sequence of actions that the misuser carries out to harm the system.	
	Steps(numbered)	Flow of events
	Postcondition	The resulting unwanted condition and the asset(s) impacted after the threat flow executes.
Specific/Bounded/Global Alternative Threat Flow	A specific alternative sequence of actions that the misuser carries out to harm the system.	
	RFS	A reference flow step number where flow branches from.
	Steps(numbered)	Flow of events
	Postcondition	The resulting unwanted condition and the asset(s) impacted after the threat flow executes.
Specific/Bounded/Global Alternative Flow	A specific alternative sequence of actions that do not result in any harm to the system.	
	RFS	A reference flow step number where flow branches from.
	Steps(numbered)	Flow of events
	Postcondition	The resulting condition after the alternative flow executes.
Mitigation Scheme	Refers to the name of the mitigation scheme, specified using our mitigation template, to mitigate this misuse case. This complements security use case(s).	

Table 2
RMCM extensions.

#	Description	Explanation
R1	MALICIOUS	Referring to a malicious actor to enforce explicitly describing the actions/steps that involve a malicious actor.
R2	DATA	Referring to the security-sensitive or privacy data to enforce explicitly describing the actions/steps that access or modify security-sensitive or privacy data.
R3	SQLI	Referring to SQL injection attacks to enforce explicitly describing the type of security threat.
R4	XPATHI	Referring to XPath injection attacks to enforce explicitly describing the type of security threat.
R5	XMLI	Referring to XML injection attacks to enforce explicitly describing the type of security threat.
R6	LDAPI	Referring to LDAP injection attacks to enforce explicitly describing the type of security threat.
R7	XSS	Referring to cross site scripting attacks to enforce explicitly describing the type of security threat.
R8	JSONI	Referring to JSON injection attacks to enforce explicitly describing the type of security threat.
R9	BO	Referring to Buffer overflow attacks to enforce explicitly describing the type of security threat.
R10	RCE	Referring to remote code execution attacks to enforce explicitly describing the type of security threat.
R11	GETS ⟨data⟩ FROM ⟨location⟩	Enforcing the explicit description of the security threats that leak data from the system (e.g., the MALICIOUS app GETS credit card DATA FROM log files).
R12	PROVIDES ⟨attack⟩ VALUES IN ⟨parameter⟩	Enforcing the explicit description of the security threat that exploits injection vulnerabilities. ⟨attack⟩ is the parameter in which the injection attack type (listed in R3-R10) is to be specified explicitly (e.g., the MALICIOUS user PROVIDES SQLI VALUES IN name and password).
R13	BYPASSES ⟨service-request⟩ REQUEST TO ⟨server-program⟩	Enforcing the explicit description of the security threats that enable a malicious actor to bypass any direct interaction with the client program and directly submit service requests to the server program (e.g., the MALICIOUS app BYPASSES view users REQUEST TO viewInfo program).
R14	EXPLOITS ⟨error-message⟩	Enforcing the explicit description of the security threat that exploits the information exposed in error or exception messages. The exposed information enables a malicious actor to understand the system better and conduct informed security attacks (e.g., the MALICIOUS user EXPLOITS exception message from the system).
R15	SENDS PRIVILEGED ⟨permission⟩ REQUEST TO ⟨client-program⟩	Enforcing the explicit description of the security threat that exploits insecure authorization schemes, which allows a malicious app to request the mobile app to execute privileged functionalities (e.g., the MALICIOUS app SENDS PRIVILEGED phone call REQUEST TO the main activity program).

describes a nominal scenario for a malicious actor to harm the system. It contains misuse case steps and a postcondition (Lines 6–13 and 38–41). A misuse case step can be one of the following interactions: a malicious actor initiates a security attack to the system (Lines 7,8,18,19,20, and 39); the system validates a request and/or data (Line 11); the system replies to a malicious actor with a result (Lines 12,17,27 and 40). A step can also capture the system altering its internal state (Lines 9 and 10). In addition, the inclusion of another use case can be specified as a step.

In RMCM, the assets impacted by a threat scenario are specified in the *Assets* field of the misuse case specifications. In addition, the assets should appear also in the postcondition of the basic threat flow or the specific/bounded/global alternative threat flows (Lines 13,23,30,41 and 47). In RMCM, the purpose of postconditions is to capture the consequences that the activities of the malicious user/app have on the assets (Line 13).

In the following, we discuss with examples how the rules in Table 2 (*R1-R15*) are applied to address *Challenge 1*:

The step in Line 7 applies *R12* to explicitly specify the security threat in which a malicious actor, tagged with the 'MALICIOUS' keyword (*R1*), initiates an SQL injection attack through the two user input fields 'user name' and 'password' of the login URL. The 'SQLI' keyword (*R3*) is used to explicitly specify the type of security threat. The step in Line 8 specifies another threat in which the malicious actor bypasses the input validation method, possibly implemented on the client side (browser), and submits the login URL to the login server program directly (*R13*). Notice that in place of the 'SQLI' keyword as the value of the parameter ⟨attack⟩ in *R12*, the keywords 'XPATHI', 'XMLI', 'LDAPI', 'XSS', 'JSONI', 'BO', 'RCE' described in *R4-R10* can be used to explicitly

```
1   MISUSE CASE Get Unauthorized Access via SQLi
2   Precondition At least one client account has already been created in the
    system.
3   Primary Actor MALICIOUS user
4   Threats Log In
5   Assets client DATA
6   Basic Threat Flow
7   1. The MALICIOUS user PROVIDES SQLI VALUES IN the user name and
       password fields of the login url.
8   2. The MALICIOUS user BYPASSES the login REQUEST TO the login
       server program.
9   3. The system builds a query with the values provided in the login url.
10  4. The system evaluates the query in the database.
11  5. The system VALIDATES THAT the query is successful.
12  6. The system SENDS the welcome message TO the MALICIOUS user.
13  Postcondition The MALICIOUS user accessed the client DATA without
                  authorization.
14  Specific Alternative Threat Flow
15  RFS 5
16  1. DO
17  2. The system SENDS the database error message DATA TO the MALICIOUS
       user.
18  3. The MALICIOUS user EXPLOITS the database error message DATA from
       the system.
19  4. The MALICIOUS user PROVIDES SQLI VALUES IN the user name and
       password fields of the login url.
20  5. The MALICIOUS user BYPASSES the login REQUEST TO the login
       server program.
21  6. UNTIL the query is successful.
22  7. RESUME STEP 6.
23  Postcondition The MALICIOUS user accessed the client DATA without
                  authorization.
24  Bounded Alternative Flow
25  RFS SATF1 1-6
26  1. IF the maximum number of login attempts is reached THEN
27  2. The system SENDS the invalid login message TO the MALICIOUS user.
28  3. ABORT.
29  4. ENDIF.
30  Postcondition The MALICIOUS user did not access the client DATA.
31  Mitigation Scheme Secure Coding for Server-side Program
32
33  MISUSE CASE Expose Information via Insecure Data Storage
34  Precondition The mobile device, in which the system is installed, also has a
             MALICIOUS app installed. The client has already used the system.
35  Primary Actor MALICIOUS app
36  Threats Get Fitter, Play Games, Do Social Activities
37  Assets user location DATA
38  Basic Threat Flow
39  1. The MALICIOUS app GETS the user location DATA FROM the log file
       of the system.
40  2. The system SENDS the user location DATA TO the MALICIOUS app.
41  Postcondition The MALICIOUS app obtained the user location DATA.
42  Specific Alternative Flow
43  RFS 2
44  1. IF the user location DATA is encrypted THEN
45  2. ABORT.
46  3. ENDIF.
47  Postcondition The MALICIOUS app did not obtain the user location DATA.
48  Mitigation Scheme Secure Coding for Mobile Program
```

Fig. 6. Misuse case specifications in RMCM.

elicit other types of code injection security threats. The step in Line 18 applies *R14* to specify a security threat that exploits the information exposed in error or exception messages, tagged with the keyword 'DATA' (*R2*). The step in Line 39 applies *R11* to specify a security threat in which a malicious actor attempts to access the user location data, tagged with the keyword 'DATA' (*R2*), locally stored in the mobile device (specified by stating the location of the data: 'log file of the system' in the ⟨location⟩ parameter). Note that in the step in Line 12, the 'welcome message' is not tagged with the keyword 'DATA' because it is not security-sensitive and thus, not considered as an information asset.

Some of the RMCM extensions in Table 2 are based on the mobile security threats listed in well accepted standards such as CWE [107] and OWASP [39], and in more general threat modeling approaches such as STRIDE from Microsoft [108]. For instance, the security extension in R15 reflects permission re-delegation threats specific to mobile apps; also local storage problem of mobile apps is covered by the extension in R11.

In the following, we discuss with examples how *Challenge 2* is addressed:

The 'VALIDATES THAT' keyword (Line 11), described in the original RUCM [15], indicates a condition that must be true to take the next step, otherwise an alternative flow is taken. It is one of the control flow structures we use for threat scenarios. In Fig. 6, the system proceeds to Step 6 (Line 12) if the query is successful (Line 11).

In the original RUCM template, there are three types of alternative flows: specific, bounded and global. In RMCM, we employ these alternative flows to describe failure scenarios for security attacks. A specific alternative flow always refers to and depends on a condition in a specific step of the basic threat flow. A bounded alternative flow refers to more than one step in the basic flow (Lines 24–30) while a global alternative flow refers to any step in the basic flow. For specific and bounded alternative flows, the keyword RFS is used to refer to one or more reference flow steps (Line 25).

In the RMCM template (Table 1), we introduce Specific/Bounded/Global alternative *threat* flows to describe alternative success scenarios and to distinguish them from failure scenarios for security attacks. For instance, in the *Get Unauthorized Access via SQLI* in Fig. 6, the specific alternative threat flow describes another success threat scenario (Lines 14–23) where the query is not validated by the system in the basic threat scenario (Line 11). The bounded alternative flow (Lines 24–30) describes the failure scenario for the attack given in this alternative threat flow (Lines 14–23).

Bounded and global alternative (threat) flows begin with the 'IF ... THEN' keyword, which is described in the original RUCM template, to describe the conditions under which alternative (threat) flows are taken (Line 26). Specific alternative flows do not necessarily begin with 'IF ... THEN' since a guard condition can be indicated in its reference flow step (Line 12). In addition, to describe threat scenarios, we also use other control flow structures — 'DO...UNTIL' and 'MEANWHILE' — which are described in the original RUCM template. For instance, in the *Get Unauthorized Access via SQLI* misuse case in Fig. 6, the malicious user tries a list of user name and password tuples iteratively in an attempt to log in the system to obtain privileges. By having such explicit loop structure (Lines 16 and 21), we are able to specify where the iteration starts and ends in the execution flow of the threat scenario.

In our previous work [16], we introduced the keyword '*SENDS ... TO*' as an RUCM extension for the system-actor interactions. The keyword eases automatic identification of steps for these interactions. For instance, Step 6 in Fig. 6 (Line 11) indicates a confirmation message from the system to the malicious user while Step 2 (Line 40) contains the user location data from the system to the malicious user.

In RMCM, use case specifications are elicited according to the original RUCM template and restriction rules [15]. Following [13], security use cases in RMCM specify the flow of events performed by the system to mitigate the attacks described in misuse cases. Differently from the original RUCM template, RMCM security use cases include two additional fields, 'Compliance' and 'Mitigate', to specify the standard provisions that the security use case should comply with and to specify the mitigated misuse case specifications (see Fig. 7).

Fig. 7 shows a simplified security use case (only some fields are shown) for mitigating the threat *Get Unauthorized Access via SQLi*. It provides compliance (Line 3) with a clause in the widely-used security standard — ISO/IEC 27001:2013 Information Security Management Systems Requirements.

Even though the proposed templates are generic, our current security extensions (Table 2) focus on the threats specific to multi-device software ecosystems including mobile and desktop devices, as per the focus of our paper. For instance, RMCM has some mobile-specific extensions in Table 2 while other extensions (e.g., SQLI in R3 in Table 2) are specific to database-centric Web applications. However, our security requirements modeling method can a priori be adapted to other types of systems. The proposed RMCM and mitigation templates in Table 1 and Table 3 are generic enough to apply our security

SECURITY USE CASE Validate Website Inputs
Precondition The system has received some inputs.
Compliance ISO/IEC 27001:2013 clause A.9.4:System and application access control.
Mitigate Get Unauthorized Access via SQLi, Expose Information via XSS, Modify Information via XSS.
Basic Flow
1. The system sanitizes the inputs according to the input specification.
2. The system VALIDATES THAT the inputs are valid.
Postcondition The system has successfully validated the inputs.
Specific Alternative Flow
RFS 2
1. The system displays an error message.
2. ABORT.
Postcondition The system has shown the invalid characters in an error message.

Fig. 7. A security use case specification.

Table 3
Mitigation template.

Scheme Name	The name of the mitigation scheme.
Brief Description	A short description about the mitigation scheme.
Actors	The actors who are responsible for reviewing and/or implementing the mitigation tasks.
Mitigated Misuse Cases	Mitigate relationships to the misuse case(s). It specifies the misuse case(s) mitigated by the mitigation scheme.
Compliance	Specifies the standard/applicable provision(s) that this mitigation scheme provides compliance.
Mitigation Tasks	Specifies the mitigation tasks. Tasks(numbered) Mitigation Tasks

requirements modeling method to other application domains by introducing further security extensions into Table 2. However, since we have only evaluated our approach in the context of multi-device software ecosystems (see Section 8), we choose to remain conservative in our conclusions.

6.3. Mitigation schemes

To address *Challenge 3*, RMCM provides the mitigation template given in Table 3. The field 'Mitigation Scheme' in misuse case specifications refers to the scheme mitigating misuse cases (Table 1). Mitigation schemes themselves are specified in a separate table, according to the mitigation template, to facilitate reuse. Differently from security use cases specifying flow of events mitigating a specific threat scenario, mitigation schemes provide the secure coding methods adopted by the system, guidelines on how to educate users and other mechanisms to prevent various security threats in general. As different security threats can be mitigated by applying standard secure coding methods, such as those listed in OWASP [39], once a mitigation scheme is specified, it can be reused or tailored as necessary for various security threats. The field 'Mitigated Misuse Cases' in Table 3 lists such various security threats mitigated by a given scheme, while the field 'Compliance' lists the standard provisions that the mitigation scheme addresses.

Mitigation schemes have a different purpose than security use cases. While a security use case describes the sequence of activities that should be performed to implement an application specific requirement, mitigation schemes aim to be more general and capture compliance with standards, regulations and guidelines. Although, in general, mitigation schemes complement security use cases, in some situations these two specifications might be used to address the same security requirements. For example, data encryption, in addition to be a mitigation scheme item (see Task Item 2 of the mitigation scheme in Fig. 8), might be modelled as a security use case. The decision to model requirements with mitigation schemes or security use cases is taken by the software engineer based on the characteristics of the developed system, according to common practices. For example, a usual practice is to adopt use cases to model significant actor-system interactions but not to model a functionality exposed by third party services or software

Scheme Name	Secure Coding for Mobile Program.
Brief Description	This mitigation scheme mitigates serious and common security threats for mobile apps.
Actors	Software Developer, Security Engineer.
Mitigated Misuse Cases	Expose Information via Insecure Data Storage, Expose Information due to Insecure Authentication.
Compliance	ISO/IEC 27001:2013 clause A.6.1.5:Information security in project management, clause A.9.4:System & application access control, clause A.10.1:Cryptographic controls.
Mitigation Tasks	1 OBFUSCATE all apk files using an Android apk obfuscator. 2 ENCRYPT sensitive data stored in mobile device, such as SQLite database, cache and log files, and SD card. 3 Apply root detection check. If jailbreak is detected, the system warns the client of potential privacy data leakage. 4 Periodically clear caching data automatically. 5 Do not grant files world readable or writable permissions. 6 Perform code integrity violation check. 7 Educate users not to download apps from unofficial stores.

Fig. 8. A sample mitigation scheme.

component interfaces (e.g., an API). According to this practice, data encryption is unlikely to be modelled as a security use case in systems that implement data encryption using standardized APIs.

Since mitigation schemes and security use cases are complementary, we have introduced the field 'Compliance' in both to provide precise traceability to specific clauses in standard provisions. Together, these artifacts provide a means for stakeholders to demonstrate compliance with applicable security and privacy standards and regulations. For instance, the mitigation scheme in Fig. 8 mitigates two misuse cases — *Expose Information via Insecure Data Storage* and *Expose Information due to Insecure Authentication*. It also supports compliance with some of the clauses in ISO/IEC 27001:2013. On our website [30], we give two additional mitigation schemes which are used to mitigate various security threats for EDLAH2.

One may argue that mitigation schemes seem to be no more than best secure coding practices with repetitions. We remark that the mitigation schemes precisely specify the actual practices adopted by the system (because not all of the security standards are applicable in practice for a specific system). Hence, they provide precise traceability to specific clauses in standard provisions for stakeholders to demonstrate compliance. In addition, mitigation schemes provide guidelines on how to educate users. As they are reusable for different security threats, repetitions would be minimal.

7. Tool support

We have implemented a tool, *RMCM-V (Restricted Misuse Case Modeling-Verifier)*, for checking the consistency between the misuse case diagram and the specifications, and the compliance of the specifications with the RMCM template. RMCM-V reports inconsistencies such as a misuse case diagram missing a *threaten* or *mitigate* relationship in specifications. Section 7.1 describes the layered architecture of the tool. Section 7.2 presents the tool features with some screenshots. For more details and accessing the tool executables, see: https://sites.google.com/site/rmcmverifier/.

7.1. Tool architecture

Fig. 9 shows the architecture of the tool. It consists of three layers: (i) *the User Interface (UI) layer*, (ii) *the Application layer*, and (iii) *the Data layer*.

User Interface (UI) Layer. This layer supports the activities of eliciting security and (mis)use cases, and mitigation schemes (see Fig. 4). We employ IBM Doors [32] for eliciting security and (mis)use case specifications and mitigation schemes according to the RUCM and RMCM templates and their restriction rules. We employ Papyrus [31]

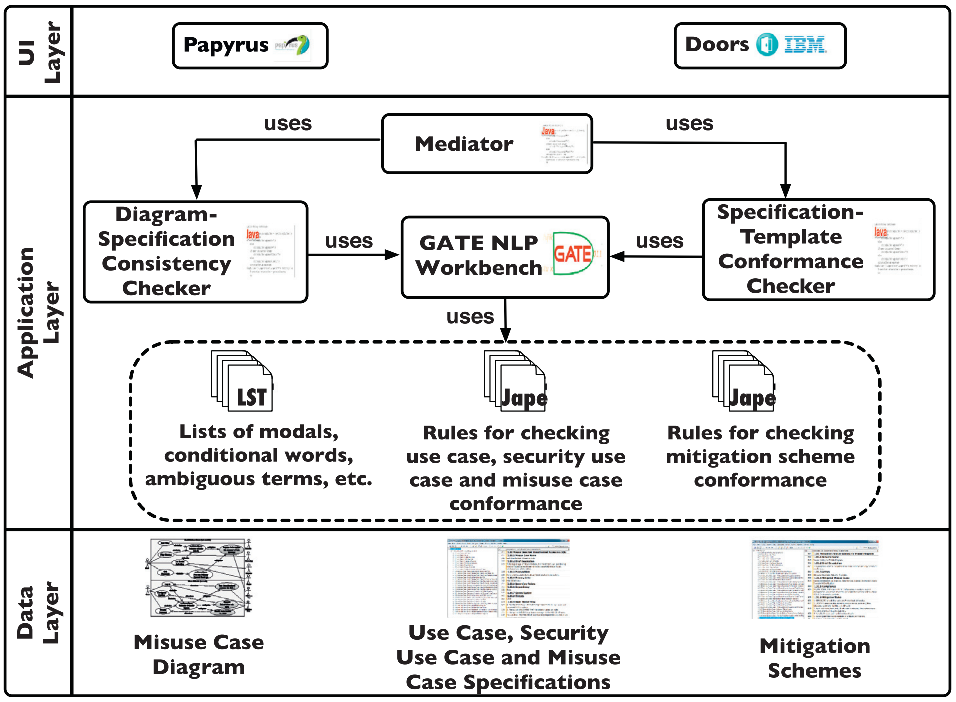

UI Layer
Papyrus
Doors IBM
Application Layer
Mediator
uses
uses
Diagram-Specification Consistency Checker
uses
GATE NLP Workbench
uses
Specification-Template Conformance Checker
uses
LST
Lists of modals, conditional words, ambiguous terms, etc.
Jape
Rules for checking use case, security use case and misuse case conformance
Jape
Rules for checking mitigation scheme conformance
Data Layer
Misuse Case Diagram
Use Case, Security Use Case and Misuse Case Specifications
Mitigation Schemes

Fig. 9. Layered architecture of RMCM-V.

for misuse case diagrams. Sindre and Opdahl [13] proposed a metamodel of the basic misuse case concepts and their relation to the UML metamodel. We adopted and implemented their proposed metamodel as a UML profile in Papyrus so that we can use the Papyrus model editor for drawing misuse case diagrams.

Application Layer. This layer supports the main activities of our modeling method in Fig. 4: *checking conformance for diagram and specifications* and *checking conformance for mitigation schemes*. It contains three main components implemented in Java: *Mediator, Diagram-Specification Consistency Checker*, and *Specification-Template Conformance Checker*. To access these *Application Layer* components through the *UI Layer*, we implemented an IBM Doors plugin.

The *Mediator* is a coordinator that manages the other two components. The *Specification-Template Conformance Checker* employs NLP to check whether the specifications and mitigation schemes comply with the RUCM and RMCM templates and their restriction rules. NLP is also used by the *Diagram-Specification Consistency Checker* to check the consistency between the misuse case diagram and specifications.

To support NLP, inspired by our previous work in requirements engineering (i.e., [18,19,109–111]), we employ a regular expression engine, called JAPE [112], in the GATE workbench [113], an open-source NLP framework. We implemented the restriction rules in JAPE. First, the specifications are split into tokens. Second, Part-Of-Speech (POS) tags (i.e., *verb, noun*, and *pronoun*) are assigned to each token. By using the restriction rules implemented in JAPE, blocks of tokens are tagged to distinguish RUCM/RMCM steps (e.g., *actor to system interaction, malicious actor to system interaction*, and *internal actions*), types of flows (i.e., *threat-specific, alternative*, and *global*), and mitigation scheme tasks. The NLP output contains the annotated use case steps and mitigation scheme tasks. The *Diagram-Specification Consistency Checker* and *Specification-Template Conformance Checker* process these annotations with the misuse case diagram to generate the list of inconsistencies among artifacts.

Data Layer. The specifications and the mitigation schemes are stored as native IBM Doors format. The misuse case diagram is stored using the UML profile mechanism.

7.2. Tool features

We describe the most important features of our tool: *managing RMCM artifacts, checking the conformance of RMCM specifications with the RMCM template, checking the consistency of the misuse case diagram and the RMCM specifications*, and *checking the conformance of mitigation schemes with the mitigation template*. These features support the steps of the modeling process given in Fig. 4.

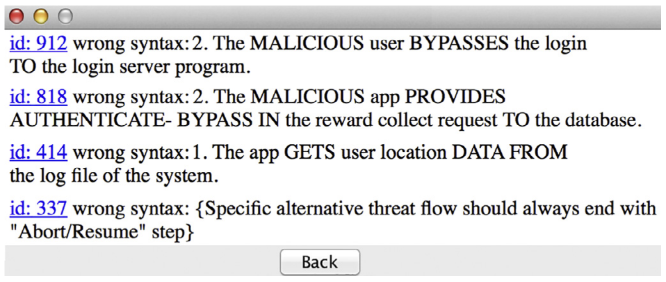
Managing RMCM artifacts. This feature supports Step 1, *Elicit Requirements as Use Cases, Security Use Cases and Misuse Cases*, and Step 3, *Elicit Mitigation Schemes for Misuse Cases*, in Fig. 4. The analyst can create, update, and delete the misuse case diagram, the corresponding specifications, and the mitigation schemes by using the selected modeling tools (i.e., IBM Doors and Papyrus) adopted in *RMCM-V*.

Checking the conformance of RMCM specifications with the RMCM template. The conformance of use case, misuse case and security use case specifications with the RMCM template and extensions needs to be ensured in Step 2, *Check Conformance for Diagram and Specifications*, in Fig. 4. Our tool automatically checks (1) if the use case and security use case specifications conform to the RUCM template [15] and (2) if the misuse case specifications conform to the RMCM template and the security extensions in Table 1 and in Table 2. Table 4 presents some of the conformance rules for misuse case specifications. For instance, a specific alternative threat flow should have the header ‘Specific Alternative Threat Flow’ followed by the ‘RFS’ keyword which refers to a misuse case step in the basic threat flow or in another alternative flow. To implement the conformance rules, RMCM-V leverages the information provided by the NLP framework; for example, in the case of the rule `'GETS ⟨data⟩ FROM ⟨location⟩'`, NLP enables RMCM-V to identify the noun phrase that corresponds with the subject of the sentence appearing in the use case step. NLP is required also to verify writing rules inherited from RUCM; for example, to foster clarity in requirements, RUCM requires that only the present tense be used and that adverbs be avoided. Both verb tenses and adverbs are determined using NLP.

Fig. 10 shows a sample result of the conformance checking of the use case and misuse case specifications for EDLAH2 in Section 6. Four types of inconsistencies are reported in Fig. 10: (i) the ‘REQUEST’

Table 4
Some of the conformance checking rules for misuse case specifications.

Modeling Element		Conformance Rules
Misuse Case Steps in Threat Flows	1	A misuse case step in a threat flow should begin with a step head containing an ordinal number and a dot punctuation.
	2	Each misuse case step should contain a structure given in one of the RMCM extensions rules R11, R12, R13, R14 and R15 in Table 2 or a structure given in the original RUCM [15].
Specific Alternative Threat Flow	1	A specific alternative threat flow should have the header 'Specific Alternative Threat Flow'.
	2	A specific alternative threat flow should begin with the 'RFS' keyword which refers to a misuse case step in the basic threat flow or in another alternative threat flow.
	3	A specific alternative threat flow should have either an 'ABORT' step or a 'RESUME' step.
	4	A specific alternative threat flow should end with a post condition.
Bounded Alternative Threat Flow	1	A bounded alternative threat flow should have the header 'Bounded Alternative Threat Flow'.
	2	A bounded alternative threat flow should begin with the 'RFS' keyword which refers to a range of misuse case steps in the basic threat flow or in another alternative threat flow.
	3	The step after the 'RFS' step in a bounded alternative threat flow should include the 'IF...THEN' keyword.
	4	The last step of a bounded alternative threat flow should have the 'ENDIF' keyword.
	5	The step before the last step of a bounded alternative threat flow should be an 'ABORT' step or a 'RESUME' step.
	6	A bounded alternative threat flow should end with a post condition.
GETS ⟨data⟩ FROM ⟨location⟩	1	The subject of the sentence should start with the 'MALICIOUS' keyword followed by the 'GETS' and 'FROM' keywords, while any string can be used to represent ⟨data⟩.
PROVIDES ⟨attack⟩ VALUES IN ⟨parameter⟩	1	The subject of the sentence should start with 'MALICIOUS' keyword followed by the 'PROVIDES' and 'VALUE IN' keywords, while any string can be used to represent ⟨attack⟩ and ⟨parameter⟩.
BYPASSES ⟨req.⟩ REQUEST TO ⟨server-program⟩	1	The subject of the sentence should start with the 'MALICIOUS' keyword followed by the 'BYPASSES' and 'REQUEST TO' keywords, while any string can be used to represent ⟨req.⟩ and ⟨server-program⟩.


id: 912 wrong syntax: 2. The MALICIOUS user BYPASSES the login TO the login server program.

id: 818 wrong syntax: 2. The MALICIOUS app PROVIDES AUTHENTICATE- BYPASS IN the reward collect request TO the database.

id: 414 wrong syntax: 1. The app GETS user location DATA FROM the log file of the system.

id: 337 wrong syntax: {Specific alternative threat flow should always end with "Abort/Resume" step}

Fig. 10. A conformance checking result reported in RMCM-V user interface.

keyword of *R13* in Table 2 is missing in the misuse case specification, (ii) there is no 'AUTHENTICATE-BYPASS' keyword which can be used with the 'PROVIDES' keyword of *R12,* (iii) the 'MALICIOUS' keyword is missing before the 'GETS' and 'FROM' keywords of *R11*, and (iv) the 'ABORT/RESUME' step is missing in one of specific alternative threat flows in the misuse case specification. By clicking the links in the user interface, the user can access the non-conformant parts of the RMCM specifications (see 'id' links in Fig. 10).

Checking the consistency of the misuse case diagram and the RMCM specifications. The consistency of the misuse case diagram and the corresponding specifications needs to be ensured as part of Step 2, *Check Conformance for Diagram and Specifications*, in Fig. 4. Table 5 presents some of the consistency rules for misuse case diagrams and RMCM specifications. For instance, for a misuse case threatening a use case in the misuse case diagram, the corresponding misuse case specification should have the threaten relation in its 'Threats' field (e.g., Lines 4 and 36 in Fig. 6). Fig. 11 presents an example output of the consistency checking of the misuse use case diagram and the corresponding specifications for EDLAH2 in Section 6.

Four types of inconsistencies are reported in Fig. 11: (i) a misuse case in the specifications does not exist in the misuse case diagram, (ii) the 'Threaten' relationship in the misuse case diagram does not exist in the specifications, (iii) the 'Threaten' relationship in the specifications does not exist in the misuse case diagram, and (iv) the 'Mitigate' relationship in the misuse case diagram does not exist in the specifications.

Checking the conformance of mitigation schemes with the mitigation template. The conformance of the mitigation schemes with the mitigation template needs to be ensured in Step 4, *Check Conformance for Mitigation Schemes*, in Fig. 4. To do so, we derived some conformance checking rules from the mitigation template in Table 3. RMCM-V provides a conformance checking report identical to Fig. 10.

8. Evaluation

The goal of our evaluation is to assess, in an industrial context and on a case study, how our proposed modeling method RMCM and our tool RMCM-V can improve the practice of eliciting and analyzing security and privacy requirements, and how well they address the challenges that we identified in Section 3. To this end, we first formulate four research questions:

- *RQ1:* Are the RMCM extensions to the RUCM template expressive enough to precisely and systematically model security threats?
- *RQ2:* Are the control flow structures of RMCM expressive enough to elicit the execution flow of threat scenarios in a structured form?
- *RQ3:* Does RMCM provide a structured way for stakeholders to specify guidance for mitigating common security threats?
- *RQ4:* Does RMCM-V provide useful automated assistance to correctly apply RMCM?

In light of the research questions given above, we evaluate our security requirements modeling method, RMCM, via reporting on (i) an industrial case study, i.e., EDLAH2, to demonstrate the feasibility of RMCM for a representative system (Section 8.1) and (ii) the results of a questionnaire survey along with discussions with EDLAH2 engineers, which aim at investigating how the approach is perceived to address the challenges listed in Section 2 and, furthermore, at gathering qualitative insights into the benefits and challenges of applying the method in an industrial setting (Section 8.2).

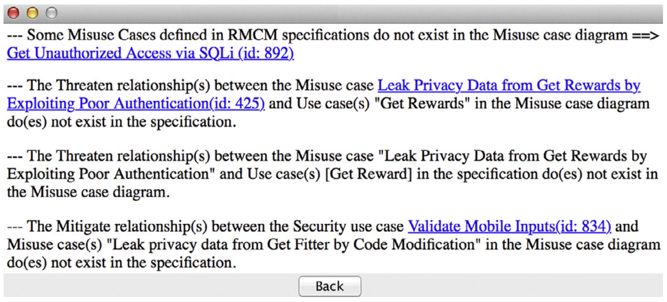
8.1. Industrial case study

We report our findings about the feasibility of our modeling method and its tool support in an industrial context. In order to experiment with RMCM in an industrial project, we applied it to the security and privacy requirements of the EDLAH2 project, which has been introduced in Section 2.

To model the security and privacy requirements of EDLAH2

Table 5
Some of the conformance checking rules for misuse case diagrams and specifications.

Modeling Element		Consistency Checking Rules
Misuse Case Name	1	A misuse case specification should have a name.
	2	A misuse case in the misuse case diagram should have a name.
	3	Each misuse case specification name should match the name of the corresponding misuse case in the misuse case diagram.
	4	Each misuse case name in the misuse case diagram should match the name of the corresponding misuse case specification.
Associations between Misusers and Misuse Cases	1	A misuser should have a name with the 'MALICIOUS' keyword in the misuse case diagram and in the misuse case specifications.
	2	A misuser in the misuse case diagram should be described as *Primary Actor* or *Secondary Actor* in the corresponding misuse case specification.
	3	In the misuse case diagram, the relation between a misuse case and its misuser should be described using the association relationship.
	4	Each relation of a misuser and a misuse case in the misuse case specification should be given in the misuse case diagram, and vice versa.
The 'Threaten' Relationship	1	For a misuse case specification with the 'Threats' field referring to a use case specification, there should be a 'Threaten' relationship from the corresponding misuse case to the corresponding use case in the misuse case diagram.
	2	For a misuse case with a 'Threaten' relationship to a use case in the misuse case diagram, the corresponding missue case specification should have the 'Threats' field referring to the corresponding use case specification.
The 'Mitigate' Relationship	1	For a security use case specification with the 'Mitigate' field referring to a misuse case specification, there should be a 'Mitigate' relationship from the corresponding security use case to the corresponding misuse case in the misuse case diagram.
	2	For a security use case with a 'Mitigate' relationship to a misuse case in the misuse case diagram, the corresponding security use case specification should have the 'Mitigate' field referring to the corresponding misuse case specification.


--- Some Misuse Cases defined in RMCM specifications do not exist in the Misuse case diagram ==> Get Unauthorized Access via SQLi (id: 892)

--- The Threaten relationship(s) between the Misuse case Leak Privacy Data from Get Rewards by Exploiting Poor Authentication(id: 425) and Use case(s) "Get Rewards" in the Misuse case diagram do(es) not exist in the specification.

--- The Threaten relationship(s) between the Misuse case "Leak Privacy Data from Get Rewards by Exploiting Poor Authentication" and Use case(s) [Get Reward] in the specification do(es) not exist in the Misuse case diagram.

--- The Mitigate relationship(s) between the Security use case Validate Mobile Inputs(id: 834) and Misuse case(s) "Leak privacy data from Get Fitter by Code Modification" in the Misuse case diagram do(es) not exist in the specification.

Fig. 11. RMCM-V user interface for reporting inconsistencies.

Table 6
The size of the RMCM artifacts in EDLAH2.

	No.	Relations	Alt. flows	Alt. threat flows	Steps	Malicious steps
Use cases	9	15	26	NA	151	NA
Security use cases	4	28	7	NA	29	NA
Mitigation schemes	3	20	NA	NA	NA	NA
Misuse cases	17	20	25	9	216	26

according to RMCM, we first examined initial EDLAH2 documentation consisting of a use case diagram and specifications, provided by the software engineers involved in the project and augmented with informal textual notes about security and privacy. Based on these artefacts, we worked together with EDLAH2 engineers to build and iteratively refine our models. EDLAH2 involves three different development teams for a total of ten software engineers. All the engineers working on EDLAH2 hold a master degree and some of them have more than ten years of software development experience. Since every team is responsible for different software components, the definition and refinement of the models has been performed independently by each team with the authors of this paper mentoring them to ensure that the methodology was applied properly. The training activity has been performed both during face-to-face project meetings and by sharing documents and tutorials in e-mails and online project meetings. Table 6 provides the size of the resulting RMCM artifacts.

In Table 6, the column 'No.' shows the numbers of use cases, security use cases, mitigation schemes, and misuse cases we modeled. The column 'Relations' shows the numbers of *include, mitigate*, and *threaten* relations among those artifacts. More precisely, in the case of the first row, the column 'Relations' indicates the number of security use cases included by functional use cases, in the case of the second and third rows, the column 'Relations' indicates the number of misuse cases mitigated by security use cases and mitigation schemes, while in the case of the fourth row the column 'Relations' indicates the number of use cases threatened by misuse cases. The columns 'Alt. flows', 'Alt. threat flows', 'Steps', 'Malicious steps' show the numbers of alternative flows, alternative threat flows, steps, and malicious steps, respectively. 'Malicious steps' denotes the steps in misuse case specifications that correspond to interactions between malicious actors and the system. 'NA' denotes "not applicable". In the following paragraphs, we rely on the data reported in Table 6 to respond to the research questions above.

RQ1

To support eliciting security threats in an explicit, precise form, RMCM includes two main extensions to RUCM, which are the identification of threaten relationships in misuse case specifications, and the adoption of specific keywords to capture common security threats.

To respond to *RQ1*, it is thus necessary to determine whether these modeling solutions (i.e., capturing threaten relationships and using security keywords) are useful, in practice, to precisely model security requirements. As an indirect measure of usefulness, we look at the number of occurrences of the threaten relationships and the security keywords in the misuse specifications of EDLAH2.

As shown in Table 6, we elicited 17 misuse cases threatening nine use cases, with a total of 20 threaten relationships among them. These numbers show that several threats tend to be relevant for each use case. This makes security requirements engineering rather complex, especially when involving many stakeholders, and it is therefore highly important to be systematic in identifying and specifying security

Table 7
Number of Occurrences of the RMCM Restrictions in EDLAH2.

R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
142	120	4	0	0	0	9	2	2	2	2	15	1	6	2

requirements.

As for restriction rules, instead, we report in Table 7 the number of times each restriction rule (*R1-R15* in Table 2) is applied when eliciting misuse cases. As shown in Table 7, we applied almost all the proposed restriction rules to systematically model the security threats of EDLAH2. Only three restriction rules (*R4, R5, R6*) were not used since EDLAH2 uses an SQL database and they correspond to security threats targeting XML and LDAP databases.

Furthermore, it is interesting to note that some of the keywords introduced by RMCM to capture common security threats were not covered in the initial EDLAH2 documentation. For instance, the extensions helped us model that the misuse case *getting unauthorized access* can be performed by means of an SQLI injection attack (*R3* and *R12*) while the misuse case *exposing information from mobile* exploits insecure data storage (*R2* and *R11*), which was not previously documented. Capturing specific threats is useful since it helps engineers in identifying mitigation mechanisms to adopt.

Finally, RMCM keywords enable the precise identification of malicious steps, i.e., misuse case steps containing information about the attack surfaces. Typical attack surfaces include parameters, URLs, files, and programs. In total, we have identified 216 steps belonging to misuse cases, and among these, we have identified 26 malicious steps. The identification of malicious steps is important because it enables engineers to easily identify attack surfaces and the mechanisms to put in place in order to prevent these attacks.

RQ2

To respond to *RQ2*, we analyze the frequency of adoption of control flow structures in the misuse case specifications of EDLAH2. More precisely, we focus on the presence of alternative threat flows, which capture the conditions and the flow of events that may still lead to successful attacks when the attacks specified in the basic threat flows fail, and alternative flows, which capture the conditions under which a potential attack does not harm the system. In addition, we report also on the frequency of the control flow keywords appearing in the specifications.

Table 6 shows that we explicitly captured nine alternative threat flows (column ‘alt. threat flows’), and 25 alternative flows (column ‘alt. flows’), thus suggesting that a security threat can materialize through multiple threat scenarios which all need to be identified and carefully analyzed. It is therefore important to have a structured and precise mechanism to express such scenarios.

Table 8 reports the number of occurrences of the RMCM Control Flow Structures in the EDLAH2 specifications. As shown in Table 8, we made frequent use of all the RMCM control flow structures in misuse cases, except the ‘MEANWHILE’ structure for concurrency sentences. This happens because in EDLAH2 we did not have to model threat scenarios in which multiple activities are executed in parallel (e.g., because the presence of two malicious users is required to put in place a specific attack). More generally, misuse cases contain the same set of keywords appearing in use cases, with the exception of the keyword ‘DO…UNTIL’ which appears in misuse cases only. This keyword is typically used to describe iterative attacks in misuse cases (e.g., malicious users trying to log into the system by trying a list of available usernames). In the use case specifications of EDLAH2, instead, we do not describe iterative behaviors of valid system users.

Table 8
Number of Occurences of the RMCM Control Flow Structures in EDLAH2.

	Misuse Cases	Security Use Cases	Use Cases	Total
DO… UNTIL	10	0	0	10
IF… THEN	15	0	8	23
VALIDATES THAT	18	8	18	44
MEANWHILE	0	0	0	0
RESUME STEP	12	0	11	23
ABORT	21	8	15	44

RQ3

To respond to *RQ3*, we focus on security use cases and mitigation schemes. Table 6 shows that we elicited four security use cases and three mitigation schemes. They typically mitigate more than one misuse case since there are 28 *mitigate* relations between security use cases and misuse cases and 20 *mitigate* relations between mitigation schemes and misuse cases (column ‘relations’). These numbers show that both security use cases and mitigation schemes can be reused across multiple misuse cases and, therefore, they are useful and reusable artifacts that should be captured independently from the misuse cases.

RQ4

To respond to *RQ4*, we applied RMCM-V to check the conformance of the first version of the EDLAH2 misuse case specifications with the RMCM template, and to check the consistency of the misuse case diagram and the (mis)use case specifications in EDLAH2. RMCM-V reported 29 warnings matching four non-conformance types when analyzing the conformance of the EDLAH2 specifications with the RMCM template (see Table 9).

Two warnings among the twenty nine warnings in the conformance checking results (2 / 29 = 6%) were related to the wrong use of the security keywords in the RMCM extensions in Table 2. All other warnings were about the violation of the rules in the original RUCM template [15], e.g., more than one action being described in a single step.

RMCM-V reported 17 inconsistencies between the misuse case diagram and the (mis)use case specifications (see Table 10). Four of them (4 / 17 = 23%) are related to the misuse cases. All other inconsistencies are about missing use cases, missing ‘Include’ relations or missing actors in use cases.

We, together with EDLAH2 engineers, were able to correct, in one iteration, all the issues reported in Table 9 and in Table 10. Our main observation was that it was easy, after some training, for the EDLAH2 engineers to correctly use our security extensions. Additionally, we manually inspected the specifications and verified that RMCM-V was able to identify all the inconsistencies and parts of the misuse case specifications that did not conform with the RMCM template.

8.2. Questionnaire study and discussions with the engineers

The questionnaire study is described and reported according to the template provided by Oppenheim [114]. To qualitatively evaluate the RMCM output in light of the four research questions presented at the beginning of this section, we had semi-structured interviews with four engineers holding various roles in the EDLAH2 consortium (i.e., project manager, software engineer, and game architect). All participants have substantial software development experience, ranging from three to 28 years. All of them had experiences with use case driven development and modeling. The interview included a presentation illustrating the RMCM steps, a tool demo, and examples from EDLAH2. The participants of the questionnaire study were also involved in the case study reported in Section 8.1. To perform the case study, we, together with the participants, had multiple face-to-face project meetings. We had shared the documents and online tutorials with them. To confirm the misuse case models of EDLAH2, we had many technical meetings with the EDLAH2 engineers, including the participants of the questionnaire.

To capture the perception of the participants regarding the potential benefits of RMCM, and assess the extent to which it addresses the targeted challenges, we handed out a questionnaire [115] including questions to be answered according to a Likert scale [114], along with open, written comments. The questionnaire was structured for the participants to assess RMCM in terms of adoption effort, expressiveness, and comparison with current practice. Table 11 shows the questions appearing in the questionnaire (divided by topic), along with the average of the scores for each answer (column result). NThe Likert scale answers provided in the questionnaire were ‘*strongly agree*’, ’*agree*’, ‘*disagree*’, and ‘*strongly disagree*’ for statement sentences in the

Table 9
Results from the analysis of non-conformant (mis)use case specifications in EDLAH2.

Non-conformance Type	Explanation	Example
Unknown Step	A flow step does not follow the restricted rules for (mis)use case steps, i.e. actor-to-actor interactions, wrong keywords, wrong structures.	- RESUME step 3 (The STEP keyword should be uppercase) - The system PROVIDES CLIENT-SENSITIVE-INFO IN the parameters sent TO the game applications (The structure of the sentence should follow the rule R12 in the RMCM extensions)
Using Adverb in a Step	An adverb is used in a flow step. This violates the rule R11 in the original RUCM [15].	IF the user name in the entered account information is already in the system THEN (The adverb 'already' appears in the sentence).
More than One Action in a Step	There are two actions in a flow step sentence. This violates the rule R4 in the original RUCM [15].	The system REQUESTS the user name and password FROM the MALICIOUS user (This sentence should be split in two steps)
Wrong Structure of Specific Alternative Flows	A specific alternative threat flow uses RFS in a wrong way at the beginning of the flow, or the last step of the flow is not a valid Abort or Resume step.	- RFS SAF 2 (It should be written in the format of 'RFS SAF 1–2'. In this case, SAF 1 points out the first specific alternative flow) - There are three specific alternative threat flows using the wrong keyword 'RESUME STEP'. - There are two specific alternative flows ending without 'ABORT' or 'RESUME STEP'.

questionnaire (e.g., question one in Table 11) while '*very probably*', '*probably*', '*probably not*', and '*surely not*' were for interrogative sentences (e.g., question 2). A discussion of the questionnaire results in light of the four research questions driving the study follows.

RQ1

The answers given to the first six questions in Table 11 indicate that the RMCM extensions provide enough expressiveness to conveniently capture security and privacy requirements in EDLAH2. More precisely, the answers to questions one and four indicate that misuse case diagrams and misuse case specifications properly support the communication between stakeholders. According to the answers to questions two and five, the participants would adopt misuse case diagrams and misuse case specifications in their daily practice. The answers to questions three and six indicate that the notation of misuse case diagrams and the use case specification template enable engineers to properly capture security requirements. The answers to questions 11, 12 and 13 further conclude that RMCM is valuable to capture security and privacy requirements.

RQ2

In our questionnaire, we did not include questions explicitly referring to control flow structures because they are perceived by engineers as a feature of the approach, which has been evaluated as being expressive enough to conveniently capture security and privacy requirements (question six in Table 11).

RQ3

The answers given to questions seven and eight let us respond to *RQ3*. The answers given to these two questions are inconsistent, with two '*agree*', one '*disagree*', and one '*strongly disagree*' for question seven (average score 2.25), and two '*probably*' and two '*probably not*' for question eight (average score 2.5). Therefore, we cannot draw clear conclusions from the data. However, we observed that the responses given by the participants are linked to their software development expertise, with the less experienced software engineers providing the more negative answers.

In general, participants find mitigation schemes less useful than misuse case specifications (the scores of the first six questions are higher). This may be due to less experienced engineers being more reluctant to document their development choices (i.e., the mitigation schemes adopted). In our context, the absence of such documentation makes it difficult to demonstrate compliance with the security and privacy standards and regulations.

RQ4

The answers given to question 14 indicate that RMCM-V provides useful assistance for minimizing inconsistencies in the RMCM artifacts of EDLAH2.

The questionnaire study had open, written comments under each section, in which the participants could state their opinions in a few sentences about how RMCM addresses the challenges reported in Section 2. Based on the initial comments, we further discussed three aspects with the participants: industrial adoption of the approach, additional extensions in RMCM, and degree of automation.

8.2.1. Industrial adoption of the approach

Given the current practice in EDLAH2, like in many other environments, there is no systematic way to capture security requirements in use case models. Even though the effort required to apply our modeling approach was considered to be reasonable by EDLAH2 engineers (questions nine and ten in Table 11), they stated that it may be a challenge to convince engineers to engage in this additional modeling effort. The costs and benefits of such an activity should be further

Table 10
Results from the analysis of the inconsistent misuse case diagram and specifications in EDLAH2.

Inconsistency Type	Explanation	Example
Lack of Security Use Cases	A security use case defined in the RMCM specifications does not exist in the misuse case diagram.	The Security Use Case 'Validate Mobile Inputs' is given in the specification, but it does not appear in the diagram.
Missing 'Include' Relations	Some 'Include' relations between use cases and security use cases in the misuse case diagram do not exist in the specifications.	Nine 'Include' relations in the misuse case diagram do not appear in the specifications. For instance, in the misuse case diagram, the use case 'Get Fitter' includes the security use case 'Provide Privacy Control Settings', but there is no relation between them in the specifications.
Missing 'Threaten' Relations	Some 'Threaten' relations between misuse cases and use cases in the misuse case diagram do not exist in the specifications, and vice versa.	The 'Threaten' relation between the misuse case 'Get Unauthorized Access via SQLi' and the use case 'Login' in the misuse case diagram does not exist in the specifications.
Missing 'Mitigate' Relations	Some 'Mitigate' relations between security use cases and misuse cases in the specifications do not exist in the misuse case diagram.	The 'Mitigate' relation between the security use case 'Provide Privacy Control Settings' and the misuse case 'Leak Privacy Data from Play Games due to Unintentional Data Flow' in the specifications does not exist in the misuse case diagram.
Missing Actor-(Mis)use Case Relations	Some Actor - (Mis)use case relations in the misuse case diagram do not exist in the specifications, and vice versa.	The relation between the use case 'Create Account' and the actor 'Manager' in the misuse case diagram does not exist in the corresponding specification.

Table 11
Questionnaire for the Evaluation of RMCM with the average of the votes.

Question	Result
Misuse Case Diagrams	
1. The diagram is simple enough to enable communication between engineers and stakeholders.	2.50
2. If a misuse case diagram like the one we presented were available to you, would you use it to help you capture or understand security threats and mitigations?	3.00
3. The notation provides enough expressiveness to conveniently capture the security threats and mitigations in your projects	3.00
Misuse Case Specifications for Capturing Security Threats.	
4. Misuse case specifications are simple enough to enable communication between engineers and stakeholders.	2.75
5. If misuse case specifications like the ones we presented were available to you, would you use those specifications to help you capture or understand security threats?	2.75
6. Security threats captured in the misuse case diagram are adequately reflected in the specifications.	3.50
Mitigation Schemes for Capturing Secure Coding Methods.	
7. Mitigation schemes are simple enough to enable communication between analysts and programmers.	2.25
8. If mitigation schemes like the ones we presented were available to you, would you use those schemes to help you capture or understand secure coding methods for mitigating security threats?	2.50
Restricted Misuse Case Modeling Method for Security and Privacy	
9. The steps in our modeling method are easy to follow.	3.50
10. The effort required to learn how to apply our method is reasonable.	2.50
11. Would you see value in adopting the presented method for capturing security threats and mitigations?	2.75
12. Does the presented method provide useful assistance for easing the communication between engineers and stakeholders?	2.25
13. Does the presented method provide useful assistance for capturing and analyzing security threats compared to the current modeling practice in your projects?	2.50
14. Do you think that the presented tool provides useful assistance for minimising the inconsistencies in misuse case diagrams and specifications?	2.75

Score for answers to interrogative questions: 4 - *very probably*, 3 - *probably*, 2 - probably not, 1 - *surely not*. Score for answers to statements: 4 - *strongly agree*, 3 - *agree*, 2 - *disagree*, 1 - *strongly disagree*.

evaluated to help with adoption. This is, however, a common and general challenge when introducing new practices in software development. For example, in the case of EDLAH2, the proposed methodology enabled the identification of effective test cases capable of identifying 14 vulnerabilities in the developed system. The test cases were derived according to traditional coverage approaches [116] that aims to generate a test case for each scenario described in the security use case and misuse case specifications.

8.2.2. Additional extensions in RMCM

The security extensions in RMCM cover various security and privacy concerns to be captured in use case models. However, EDLAH2 engineers stated that, due to rapidly changing software and hardware technology, new types of security threats will likely need to be covered with further security extensions. In a way, such extensions can be treated as a knowledge repository of potential vulnerabilities and their associated mitigation schemes. Such repository has to be regularly updated and is expected to help creating awareness of security threats and solutions across an organization.

8.2.3. Degree of automation

RMCM consists of various automated security requirements modeling and specification activities in the context of use case-driven development. Though modeling security requirements in misuse case models is mostly manual, RMCM-V provides automatic consistency checking for these models and feedback to the analyst to help them refine and correct the models. EDLAH2 engineers considered the automated consistency checking of RMCM artifacts to be highly valuable.

8.3. Threats to validity

The main threat to the validity of our evaluation is the generalizability of the conclusions. To mitigate the threat, we applied RMCM to a representative system that includes nontrivial use cases in an application domain entailing numerous and varied security threats. Although we had a relatively low number of respondents in our interviews, we selected the respondents to hold various roles and with substantial industry experience. To limit threats to the internal validity of the case study, we had many meetings with the EDLAH2 engineers to verify the correctness and completeness of our models.

9. Conclusion

This paper presents a use case-driven security requirements modeling method, called RMCM, for documenting the security and privacy requirements of multi-device software ecosystems in a structured and analyzable form. Our main motivation is to enable security and privacy requirements modeling by relying on commonly used artifacts in use-case driven development and by adding a limited number of extensions, thus achieving widespread applicability.

RMCM builds on and integrates existing work and is supported by a tool employing NLP for checking the consistency of artifacts and compliance to the RMCM templates. The key characteristic of our method is that it captures threat scenarios and mitigation schemes in an explicit and structured form, thus enabling both automated analysis of threat scenarios, e.g., consistency and conformance checking, and reuse of mitigation schemes. Initial results from structured interviews with experienced engineers suggest that RMCM is precise and practical to capture the security and privacy requirements of multi-device software ecosystems in industrial settings.

In addition to supporting more precise and complete security requirements, RMCM is a first step to achieve a longer-term objective: automated test generation for requirements-driven security testing. Our plan for the next stages is to provide an automated technique that generates security test cases from misuse case models. Our ultimate objective is to achieve adequate coverage of the specified security and privacy requirements, with traceability information between security requirements and generated test cases.

Acknowledgment

This work is supported by the National Research Fund, Luxembourg FNR/P10/03, INTER/AAL/15/11213850, and INTER/DFG/14/11092585, and by the European Research Council (ERC) under the

European Union's Horizon 2020 research and innovation programme (grant agreement No 694277).

References

[1] USC Credit Union, System providing home-banking. [Online]. Available:https://member.usccreditunion.org/.
[2] Spotify, System providing music streaming software and services. [Online]. Available: https://www.spotify.com.
[3] DeliveryHero, System providing food delivery software and services, Visited in 2017. [Online]. Available: https://www.deliveryhero.com/.
[4] FitBit, System providing personal training software services, 2017[Online]. Available: https://www.fitbit.com.
[5] A.K. Jain, D. Shanbhag, Addressing security and privacy risks in mobile applications, IT Professional 14 (5) (2012) 28–33.
[6] A. Bortz, D. Boneh, Exposing private information by timing web applications, WWW'07, (2007), pp. 621–628.
[7] C. Larman, Applying UML and Patterns:An Introduction to Object-Oriented Analysis and Design and the Unified Process, Prentice Hall Professional, 2002.
[8] J. McDermott, C. Fox, Using abuse case models for security requirements analysis, ACSAC'99, (1999).
[9] J. McDermott, Abuse-case-based assurance arguments, ACSAC'01, (2001).
[10] D.G. Firesmith, Security use cases, Journal of Object Technology 2 (3) (2003) 53–64.
[11] A.L. Opdahl, G. Sindre, Experimental comparison of attack trees and misuse cases for security threat identification, Information and Software Technology 51 (2009) 916–932.
[12] L. Rostad, An extended misuse case notation: Including vulnerabilities and the insider threat, REFSQ'06, (2006), pp. 33–43.
[13] G. Sindre, A.L. Opdahl, Eliciting security requirements with misuse cases, Requirements Engineering 10 (2005) 34–44.
[14] G. Sindre, A.L. Opdahl, Templates for misuse case description, REFSQ'01, (2001).
[15] T. Yue, L.C. Briand, Y. Labiche, Facilitating the transition from use case models to analysis models: Approach and experiments, ACM Transactions on Software Engineering and Methodology 22 (1) (2013) 1–38.
[16] C. Wang, F. Pastore, A. Goknil, L.C. Briand, M.Z.Z. Iqbal, Automatic generation of system test cases from use case specifications, ISSTA'15, (2015), pp. 385–396.
[17] C. Wang, F. Pastore, A. Goknil, L.C. Briand, M.Z.Z. Iqbal, UMTG: a toolset to automatically generate system test cases from use case specifications, ESEC/FSE'15, (2015), pp. 942–945.
[18] I. Hajri, A. Goknil, L.C. Briand, T. Stephany, Applying product line use case modeling in an industrial automotive embedded system: Lessons learned and a refined approach, MODELS'15, (2015), pp. 338–347.
[19] I. Hajri, A. Goknil, L.C. Briand, T. Stephany, Configuring use case models in product families, Software and Systems Modeling (2016).
[20] I. Hajri, A. Goknil, L.C. Briand, T. Stephany, PUMConf: a tool to configure product specific use case and domain models in a product line, FSE'16, (2016), pp. 1008–1012.
[21] I. Hajri, A. Goknil, L.C. Briand, T. Stephany, Incremental reconfiguration of product specific use case models for evolving configuration decisions, REFSQ'17, (2017), pp. 3–21.
[22] I. Hajri, A. Goknil, L.C. Briand, A change management approach in product lines for use case-driven development and testing, REFSQ Workshops, (2017).
[23] I. Hajri, A. Goknil, L.C. Briand, T. Stephany, Change impact analysis for evolving configuration decisions in product line use case models, Journal of Systems and Software (2018).
[24] M. Hansen, M. Jensen, M. Rost, Protection goals for privacy engineering, SPW'15, (2015), pp. 159–166.
[25] D.J. Solove, A taxanomy of privacy, University of Pennsylvania Law Review 154 (3) (2006) 477–560.
[26] A. Pfitzmann, M. Hansen, A Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management, Technical Report, TU Dresden, 2010.
[27] ISO-IEC, ISO/IEC 29100:2011 information technology - security techniques - privacy framework.
[28] OECD, OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, Technical Report, Organisation of Economic Co-Operation and Development, 1980.
[29] US Federal Trade Commission, Privacy online: A report to congress, https://www.ftc.gov/sites/default/files/documents/reports/privacy-online-report-congress/priv-23a.pdf.
[30] X.P. Mai, RMCM-V: a tool for checking consistencies between misuse case diagram, specifications, and restricted misuse case modeling templates, 2017, https://sites.google.com/site/rmcmverifier/.
[31] Papyrus, https://www.eclipse.org/papyrus.
[32] IBM Doors, http://www.ibm.com/software/products/ca/en/ratidoor.
[33] EDLAH2: Active and Assisted Living Programme, http://www.aal-europe.eu/projects/edlah2/.
[34] S. Deterding, D. Dixon, R. Khaled, L. Nacke, From game design elements to gamefulness: Defining "gamification", MindTrek'11, ACM, 2011, pp. 9–15.
[35] iCare, http://www.icare247.eu/edlah2/.
[36] A. Cockburn, Writing effective use cases, Addison-Wesley, 2001.
[37] F. Armour, G. Miller, Advanced Use Case Modeling: Software Systems, Addison-Wesley, 2001.
[38] D. Kulak, E. Guiney, Use Cases: Requirements in Context, Addison-Wesley, 2003.
[39] OWASP, OWASP Top 10 Mobile Security Risks, https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10.
[40] OWASP Top 10 Web Security Risks, https://www.owasp.org/index.php/Top_10_2013-Top_10.
[41] S. Turpe, The trouble with security requirements, RE'17, (2017), pp. 122–133.
[42] B. Fabian, S. Gurses, M. Heisel, T. Santen, H. Schmidt, A comparison of security requirements engineering methods, Requirements Engineering 15 (2010) 7–40.
[43] D. Mellado, C. Blanco, L.E. Sanchez, E. Fernandez-Medina, A systematic review of security requirements engineering, Computer Standards & Interfaces 32 (2010) 153–165.
[44] A. Souag, R. Mazo, C. Salinesi, I. Comny-Wattiau, Reusable knowledge in security requirements engineering: a systematic mapping study, Requirements Engineering 21 (2016) 251–283.
[45] P. Salini, S. Kanmani, Survey and analysis on security requirements engineering, Computers and Electrical Engineering 38 (2012) 1785–1797.
[46] I.A. Tondel, M.G. Jaatun, P.H. Meland, Security requirements for the rest of us: A survey, IEEE Software 25 (1) (2008) 20–27.
[47] P. Anthonysamy, A. Rashid, R. Chitchyan, Privacy requirements: present & future, ICSE-SEIS'17, (2017), pp. 13–22.
[48] K. Beckers, Comparing privacy requirements engineering approaches, ARES'12, (2012), pp. 574–581.
[49] S. Gurses, B. Berendt, T. Santen, Multilateral security requirements analysis for preserving privacy in ubiquitous environments, UKDU'06, (2006).
[50] S. Gurses, T. Santen, Contextualizing security goals—a method for multilateral security requirements elicitation, Sicherheit'06, (2006), pp. 42–53.
[51] N.R. Mead, E.D. Hough, T.R. Stehney, Security Quality Requirements Engineering (SQUARE) Methodology, CMU/SEI-2005-TR-009, Carnegie Mellon Software Engineering Institute, 2005.
[52] T. Lodderstedt, D.A. Basin, J. Doser, SecureUML: A UML-based modeling language for model-driven security, UML'02, (2002), pp. 426–441.
[53] J. Jürjens, Secure Systems Development with UML, Springer, 2003.
[54] L. Liu, E. Yu, J. Mylopoulos, Security and privacy requirements analysis within a social setting, RE'03, (2003), pp. 151–161.
[55] G. Elahi, E. Yu, A goal oriented approach for modeling and analyzing security trade-offs, ER'07, (2007), pp. 375–390.
[56] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, Modeling security requirements through ownership, permission and delegation, RE'05, (2005), pp. 167–176.
[57] A. van Lamsweerde, Elaborating security requirements by construction of intentional anti-models, ICSE'04, (2004), pp. 148–157.
[58] H. Mouratidis, P. Giorgini, Secure tropos: a security-oriented extension of the tropos methodology, International Journal of Software Engineering and Knowledge Engineering 17 (2) (2007) 285–309.
[59] L. Pasquale, P. Spoletini, M. Salehie, L. Cavallaro, B. Nuseibeh, Automating trade-off analysis of security requirements, Requirements Engineering 21 (4) (2016) 481–504.
[60] C. Kalloniatis, E. Kavakli, S. Gritzalis, Addressing privacy requirements in system design: The pris method, Requirements Engineering 13 (3) (2008) 241–255.
[61] L. Lin, B. Nuseibeh, D. Ince, M. Jackson, Using abuse frames to bound the scope of security problems, RE'04, (2004), pp. 354–355.
[62] D. Hatebur, M. Heisel, H. Schmidt, Security engineering using problem frames, ETRICS'06, (2006), pp. 238–253.
[63] D. Hatebur, M. Heisel, H. Schmidt, Analysis and component-based realization of security requirements, AReS'08, (2008), pp. 195–203.
[64] C.B. Haley, R. Laney, J.D. Moffett, B. Nuseibeh, Security requirements engineering: A framework for representation and analysis, IEEE Transactions on Software Engineering 34 (1) (2008) 133–153.
[65] C.B. Haley, R. Laney, J.D. Moffett, B. Nuseibeh, Picking battles: the impact of trust assumptions on the elaboration of security requirements, iTrust'04, (2004), pp. 347–354.
[66] K. Thomas, A.K. Bandara, B.A. Price, B. Nuseibeh, Distilling privacy requirements for mobile applications, ICSE'14, (2014), pp. 871–882.
[67] F. den Braber, I. Hogganvik, M.S. Lund, K. Stolen, F. Vraalsen, Model-based security analysis in seven steps — a guided tour to the CORAS method, BT Technology Journal (2007) 101–117.
[68] Y. Asnar, P. Giorgini, F. Massacci, N. Zannone, From trust to dependability through risk analysis, ARES'07, (2007), pp. 19–26.
[69] A. Cailliau, A. van Lamsweerde, Assessing requirements-related risks through probabilistic goals and obstacles, Requirements Engineering 18 (2) (2013) 129–146.
[70] A. van Lamsweerde, Requirements Engineering: from System Goals to UML Models to Software Specifications, John Wiley and Sons, 2009.
[71] Y. Asnar, P. Giorgini, J. Mylopoulos, Goal-driven risk assessment in requirements engineering, Requirements Engineering 16 (2011) 101–116.
[72] N. Mayer, E. Dubois, A. Rifaut, Requirements engineering for improving business/it alignment in security risk management methods, Enterprise Interoperability II, (2007), pp. 15–26.
[73] Common Criteria for Information Technology Securitys Evaluation, 2006, http://www.commoncriteriaportal.org.
[74] D. Mellado, E. Fernandez-Medina, M. Piattini, A comparison of the Common Criteria with proposals of information systems security requirements, ARES'06, (2006), pp. 654–661.
[75] D. Mellado, E. Fernandez-Medina, M. Piattini, Applying a security requirements engineering process, ESORICS'06, (2006), pp. 192–206.
[76] K. Rannenberg, A. Pfitzmann, G. Müller, IT security and multilateral security,

Multilateral Security in Communications–Technology, Infrastructure, Economy (1999) 21–29.
[77] S. Spiekermann, L.F. Cranor, Engineering privacy, IEEE Transactions on Software Engineering 35 (1) (2009) 67–82.
[78] M. Jackson, Problem Frames: Analysing and Structuring Software Development Problems, Addison-Wesley, 2001.
[79] I. Alexander, Misuse cases: Use cases with hostile intent, IEEE Software 20 (1) (2003) 58–66.
[80] I. Alexander, Misuse cases help to elicit non-functional requirements, Computing & Control Engineering Journal 14 (1) (2003) 40–45.
[81] I. Alexander, Initial industrial experience of misuse cases in trade-off analysis, RE'02, (2002), pp. 61–70.
[82] D.G. Rosado, E. Fernandez-Medina, J. Lopez, Applying a UML extension to build use cases diagrams in a secure mobile grid application, ER'09 Workshops, (2009), pp. 126–136.
[83] G. Sindre, A.L. Opdahl, G.F. Brevik, Generalization/specialization as a structuring mechanism for misuse cases, SREIS'02, (2002).
[84] G. Sindre, D.G. Firesmith, A.L. Opdahl, A reuse-based approach to determining security requirements, REFSQ'03, (2003).
[85] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, W. Joosen, A privacy threat analysis framework: Supporting the elicitation and fulfillment of privacy requirements, Requirements Engineering 16 (1) (2011) 3–32.
[86] I. Omoronyia, M. Salehie, R. Ali, H. Kaiya, B. Nuseibeh, Misuse case techniques for mobile privacy, PriMo'11, (2011).
[87] M. El-Attar, Towards developing consistent misuse case models, Journal of Systems and Software 85 (2) (2012) 323–339.
[88] M. El-Attar, Using SMCD to reduce inconsistencies in misuse case models: A subject-based empirical evaluation, Journal of Systems and Software 87 (2014) 104–118.
[89] A. van Lamsweerde, Elaborating security requirements by construction of intentional anti-models, ICSE'04, (2004), pp. 148–157.
[90] F. Swiderski, W. Snyder, Threat Modeling, Microsoft Press, Redmond, WA, USA, 2004.
[91] A. Rashid, S.A.A. Naqvi, R. Ramdhany, M. Edwards, R. Chitchyan, M.A. Babar, Discovering unkown known security requirements, ICSE'16, (2016), pp. 866–876.
[92] B. Glaser, A. Strauss, The Discovery of Grounded Theory, Aldine Publishing Co., 1967.
[93] C. Johnson, A Handbook of Accident and Incident Reporting, Glasgow University Press, 2003.
[94] J.I. Hong, J.D. Ng, S. Lederer, J.A. Landay, Privacy risk models for designing privacy-sensitive ubiquitous computing systems, DIS'04, (2004), pp. 91–100.
[95] T.D. Breaux, H. Hibshi, A. Rao, Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements, Requirements Engineering 19 (3) (2014) 281–307.
[96] T.D. Breaux, A. Rao, Formal analysis of privacy requirements specifications for multi-tier applications, RE'13, (2013), pp. 14–23.
[97] J. Whittle, D. Wijesekera, M. Hartong, Executable misuse cases for modeling security concerns, ICSE'08, (2008), pp. 121–130.
[98] G. Sindre, Mal-activity diagrams for capturing attacks on business processes, REFSQ'07, (2007), pp. 355–366.
[99] E. Song, R. Reddy, R. France, I. Ray, G. Georg, R. Alexander, Verifiable composition of access control and application features, SACMAT'05, (2005), pp. 120–129.
[100] J. Jürjens, Towards development of secure systems using umlsec, FASE'01, (2001), pp. 187–200.
[101] B. Schneier, Modelling security threats, Dr. Dobb's Journal (1999).
[102] L. Lin, B. Nuseibeh, D. Ince, M. Jackson, J. Moffett, Introducing abuse frames for analysing security requirements, RE'03, (2003), pp. 371–372.
[103] J. Großmann, F. Seehusen, Combining security risk assessment and security testing based on standards, RISK'15, (2015), pp. 18–33.
[104] Etsi-eg-203-251: Methods for testing & specification; risk-based security assessment and testing methodologies, 2015.
[105] Y.-G. Kim, S. Cha, Threat scenario-based security risk analysis using use case modeling in information systems, Security and Communication Networks 5 (3) (2012) 293–300.
[106] CVSS: Common Vulnerability Scoring System, 2018, https://www.first.org/cvss/.
[107] CWE/SANS Top 25 Most Dangerous Software Errors, http://cwe.mitre.org/top25/.
[108] The STRIDE Threat Model, https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx.
[109] C. Arora, M. Sabetzadeh, L.C. Briand, F. Zimmer, Automated checking of conformance to requirements templates using natural language processing, IEEE Transactions on Software Engineering 41 (10) (2015) 944–968.
[110] C. Arora, M. Sabetzadeh, A. Goknil, L.C. Briand, F. Zimmer, Change impact analysis for natural language requirements: An nlp approach, RE'15, (2015), pp. 6–15.
[111] C. Arora, M. Sabetzadeh, A. Goknil, L.C. Briand, F. Zimmer, NARCIA: an automated tool for change impact analysis in natural language requirements, ESEC/FSE'15, (2015), pp. 962–965.
[112] H. Cunningham, et al. Developing language processing components with gate version 8 (a user guide), http://gate.ac.uk/sale/tao/tao.pdf.
[113] The GATE workbench, http://gate.ac.uk/.
[114] A.N. Oppenheim, Questionnaire Design, Interviewing and Attitude Measurement, Continuum, 2005.
[115] [Online]. Available: https://goo.gl/forms/OrAZcZLvsVm5tUN13.
[116] I. Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.