Towards Risk-Driven Security Requirements Management in Agile Software Development

Dan Ionita[1(✉)], Coco van der Velden[2], Henk-Jan Klein Ikkink[2], Eelko Neven[2], Maya Daneva[1], and Michael Kuipers[2]

[1] Department Cybersecurity and Safety, University of Twente, Enschede, The Netherlands
{d.ionita,m.daneva}@utwente.nl
[2] Centric B.V., Gouda, The Netherlands
{coco.van.der.velden,henk-jan.klein.ikkink,eelko.neven, michael.kuipers}@centric.eu
https://scs.ewi.utwente.nl/

Abstract. The focus on user stories in agile means non-functional requirements, such as security, are not always explicit. This makes it hard for the development team to implement the required functionality in a reliable, secure way. Security checklists can help but they do not consider the application's context and are not part of the product backlog.

In this paper we explore whether these issues can be addressed by a framework which uses a risk assessment process, a mapping of threats to security features, and a repository of operationalized security features to populate the product backlog with prioritized security requirements. The approach highlights the relevance of each security feature to product owners while ensuring the knowledge and time required to implement security requirements is made available to developers. We applied and evaluated the framework at a Dutch medium-sized software development company with promising results.

Keywords: Secure software development · Security requirements · Risk assessment · Empirical research method

1 Introduction

Agile software development relies on the team's ability to decompose, refine, and operationalize high-level user requirements such as user stories. The majority of users and customers lack awareness of the security risks in the implementation and usage of the software and settle for compliance. Furthermore, most agile

Supported by The Netherlands Organisation for Scientific Research (NWO) in the context of cyber-security research (grant number 628.001.011).

C. Cappiello and M. Ruiz (Eds.): CAiSE Forum 2019, LNBIP 350, pp. 133–144, 2019.
https://doi.org/10.1007/978-3-030-21297-1_12

teams do not have a security expert on board. Therefore, product owners (POs) have a hard time identifying and prioritizing security requirements and developers often rely on security checklists which are not integrated with agile project management processes and tools. This results in software which fails to properly mitigate many risks relevant to its users or its application domain.

To address this, we propose a secure software development framework consisting of (1) a high-level risk assessment process to be undertaken with the application's stakeholders, (2) a mapping of threats to security requirements and (3) a searchable repository of security requirements integrated with agile project management tools. The goal of the risk assessment is to identify and prioritize risks, the mapping is used to derive high level security requirements (i.e. features) based on these risks, and the repository makes it easy to inject these features together with their operationalizations into the agile development workflow. The three components work together to support risk-driven selection, prioritization and implementation of security requirements in a way that requires minimal security knowledge and effort from users, customers, and POs. As a whole, the approach is designed to provide traceability of security requirements and forces the development team to consider the effort required to implement security features in their planning.

The framework aims to align with the agile manifesto [3]. First, it provides a means to discuss security with the customer of an agile project. Second, it produces a list of prioritized requirements in a format that can be directly imported into product backlogs. Third, it leaves it up to the teams to break down the requirements into tasks and add these to sprints. Fourth, it includes control points for testing the implementation. Fifth, it is able to respond to changes in the risk landscape. Furthermore, the framework was applied in a real-life organization and its evaluation shows promising results.

In what follows, Sect. 2 summarizes our research methodology. Sections 3 and 4 position our work in relation to the real-world problems that we want to address as well as related publications. Section 5 introduces our framework based on a practical example of how this was implemented at Centric, a medium-sized software development company. Section 6 discusses our preliminary evaluation and Sect. 7 draws wraps up with conclusions and future work.

2 Research Methodology

This paper is the result of an extensive collaboration between the University of Twente and Centric B.V., a medium-sized Dutch application provider. Therefore, the underlying methodology applied throughout the research is Technical Action Research (TAR): a technique is designed and applied to a real-life problem in order to draw conclusions about both the technique and the problem. TAR helps both the company, in that it is provided with a working solution, and the researcher, in that he/she has the opportunity to perform real-life validation [20].

Problem Investigation. To better understand the problem context, we did a literature survey on security requirements management in agile software development. In parallel, we interviewed security coordinators, penetration testers, developers, POs, as well as the management and governance team working on various projects within Centric. The findings are summarized in the Sect. 3.

Treatment Design. To define our solution direction, we explored literature on strategies for incorporating security in agile workflows. The findings are summarized in Sect. 4. We designed a framework which aligns with some of these strategies while addressing the issues highlighted by the problem investigation. Each component of the framework was developed iteratively in consultation with relevant stakeholders from Centric. The framework is described in Sect. 5.

Treatment Validation. To validate the proposed treatment we ran five focus groups within Centric where we applied the risk assessment methodology described in Sect. 5.1. We systematically compared the results of applying the mapping described in Sect. 5.2 to the judgment of security experts. Finally, we created a template project according to the repository structure proposed in Sect. 5.3 in a major issue tracking software and ran a survey to assess its usability and utility.

3 Motivation and Background

Established approaches to security engineering as part of software development fail to address the particular needs of agile [2]. As a result, security considerations in agile software development are often based on security baselines, despite the fact that best practice insists security should be risk-driven [1,4,8]. "Discrete techniques" such as security checklists integrate very poorly into agile approaches [18]. The requirements listed in baselines and checklists don't always have an owner and are often considered towards the end of development [19].

Daneva and Wang [5] indicate that security requirements engineering in agile boils down to documenting risks and mitigations. But POs, as well as developers, often lack security knowledge [19]. Our framework provides assistance in formulating risk scenarios and centralizes security knowledge, making it easily reusable across teams and even across projects.

Daneva and Wang also point out that the gate-keeping role of the PO often hampers the elicitation and implementation of security requirements. This is mainly because agile is business-value-driven [3] and security is hard to "sell" [6]. We propose mitigating this by highlighting the business value – in terms of risk reduction – that each security requirements provides.

Furthermore, our informal interviews and conversations with domain experts and stakeholders revealed that:

- Security requirements are not risk-based. We mitigate this by maintaining a mapping base of risks and requirements.

- Security requirements are not user-driven. We mitigate this by prescribing a non-technical risk assessment methodology.
- Security requirements are not application-specific. We mitigate this by linking requirements to the result of the risk assessment.
- Penetration tests are standardized. We mitigate this by generating a list of the most important security requirements to be tested.
- Security requirement documents are missing implementation-specifics. We mitigate this by maintaining a central repository of operationalized security features.
- It's hard to keep track of the implementation status of security requirements. We mitigate this by making security requirements available on issue trackers.

4 Related Work

Siponen et al. [18] conclude that an agile security requirements management technique must include a quantified risk assessment in the requirements analysis phase. Security requirements should be explicitly included in the design phase, their implementation must be monitored throughout the implementation phase and they should be tested in the testing phase. Our proposal mandates starting with a risk assessment, including the resulting requirements in the planning and effort estimations, and making sure acceptance criteria are known and tested.

A notable approach for eliciting security requirements in agile are *abuser stories*. Similar to abuse cases, they document threat scenarios [14,15]. However, this approach has some caveats. E.g. several threat scenarios can be mitigated by the same security feature, sometimes with different efficiency. And since most issue trackers do not allow a backlog item to have multiple parents, defining these as backlog items results in many security requirements being duplicated. This would also fail to reflect the relative importance of security features with regard to their efficacy and the number of threat scenarios they mitigate. In order to avoid these issues, our approach maintains a separate overview of threat scenarios as part of the risk assessment document described in Sect. 5.1.

Terpstra et al. [19] performed a practitioner survey of problems and coping strategies for handling security requirements in agile project management and agile software development projects. The methodology proposed in this paper aligns well with many of these strategies. For example, by adding security features – including acceptance criteria - to the backlog we are making sure they are considered during effort estimations, and that these features are part of the definition of "done".

5 The Proposed Framework

The aim of the framework is to support the identification, prioritization and implementation of security requirements in an agile workflow. Its application consists of three phases:

1. **Risk assessment:** Each risk is quantified and mapped to one or more of threats in consultation with relevant stakeholders (the left side of Fig. 1). This performed according to a pre-defined *risk assessment methodology* described in Sect. 5.1 below.
2. **Prioritization of security requirements:** Based on the ranked list of threats resulting from the risk assessment, a prioritized list of security requirements is automatically derived (the middle part of Fig. 1). This is achieved by means of an intermediary *threat-requirement map* described in Sect. 5.2 below.
3. **Populate product backlog:** The PO imports the relevant security requirements with associated priorities to the product backlog of his agile issue tracking tool of choice (the right side of the Fig. 1). This is facilitated by means of a *security requirements repository* described in Sect. 5.3 below.

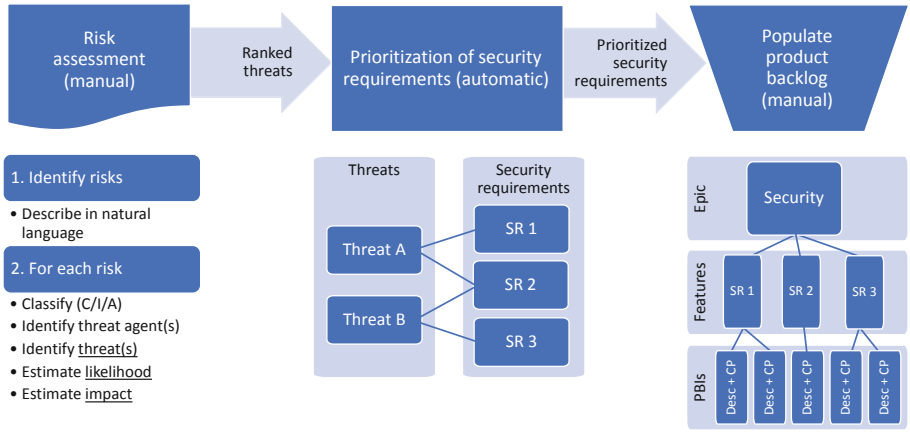

Fig. 1. Overview of the proposed framework

5.1 The Risk Assessment Methodology

Most secure software development guidelines such as the CIP Overheid [12] or the ISO 27000x series [9–11] recommend performing a risk or threat assessment as early in the software lifecycle as possible. This helps avoid architectural risks and reduces the amount of work needed to fix security issues late in the project. But more importantly, it provides a good understanding of the most significant threats and risks.

In our framework, risk assessment serves as a starting point. Its results are to be correlated with the mapping described in the following section in order to produce a ranked list of security requirements. To this end, the risk assessment must be correct and complete, and should therefore be performed in close

consultation with the PO and customer in order to make sure the most relevant risks are identified, that the impact estimations are accurate, and that the resulting mitigations are taken into consideration during agile planning. In addition, the assessment must produce output which can be consumed by the threat-requirement map, namely a quantitative ranked list of pre-defined threats.

Structure. We use a spreadsheet as the basis for our assessment, see Fig. 2. It consists of the following columns:

Risk label - a brief description of the risk.
Explanation - an description of the process by which the risk could materialize.
Type - Confidentiality, Integrity, or Availability.
Threat agent(s) - An individual or group which are likely to try to materialize the risk.
Threat(s) - One or more cyber-threats by which the risk could materialize.
Likelihood - The expected frequency with which the Threat agent would attempt to use the Threat in order to materialize the risk.
Impact - The cost or loss caused by the occurrence of the risk.
Rating - Likelihood x Impact.

Risk label	Explanation	Type C/I/A	Threat agent(s)	Threat(s)	Likelihood (0-100)	Impact (0-100)
Expose data of famous people	Employee accidentaly leaks personal (sensitive) information of Politicians or celeberties. E.g.HR reviews, medical info, leave.	Confidentiality	Insiders	Information leakage	50	50
Data breach - famous people	Hacker beach the system. Politicians, Royal and celeberties have personal (sensitive) exposed. E.g. HR reviews, medical info, leave.	Confidentiality	Cyber-criminals	Data breach	20	55
Hacktivist leak sensitive info	Hackivist - targeted attack e.g. salary data of politicians and charities, to expose controversial information.	Confidentiality	Hacktivists	Data breach, Phishing	5	65
Fraudulent contract adjustments	HR contracts are adjusted for personal benefit. E.g. salary, or contract hours.	Integrity	Insiders	Insider threat, Web-based attack	10	15

Fig. 2. Fragment of a risk assessment

Process. The table (Fig. 2) is filled in from left to right, however, we found that first selecting a likely threat agent stimulates creativity. This is in line with the philosophy of the Intel's Threat Agent Risk Assessment [17] which starts by agreeing on a list of relevant threat agents. Each risk is given a label, described in free text, and classified in terms as confidentiality, integrity, or availability. In order to further scope down the risk and ensure consensus among participants, a relevant threat agent(s) is chosen if one was not chosen already. It is possible that the same risk produces a different impact, or manifests with a different likelihood depending on the threat agent and their purpose. Therefore, the same risk may appear on multiple rows, but mapped to a different threat agent. Then, each risk is mapped to one or more of the pre-defined threats. Finally, each risk is quantified in terms of likelihood and impact which are multiplied in order to obtain a risk rating.

5.2 The Threat-Feature Map

To arrive at the ranked list of security requirements needed to populate the product backlog, a mapping between the threat taxonomy used in the risk assessment and a set of security features is necessary. To strengthen the usability and justifiability of the mapping, the list of threats should be based on an established threat taxonomy such as ENISA's [7] or Intel's [17], and the list of requirements should be based on established secure software development guidelines such as OAWSP [16] or Grip on SSD [13]. Note that there is a many-to-many relationship between threats and requirements. Furthermore, this relationship is not binary; some security requirements are better at mitigating a threat than others. This *relevance factor* should also be reflected in the mapping.

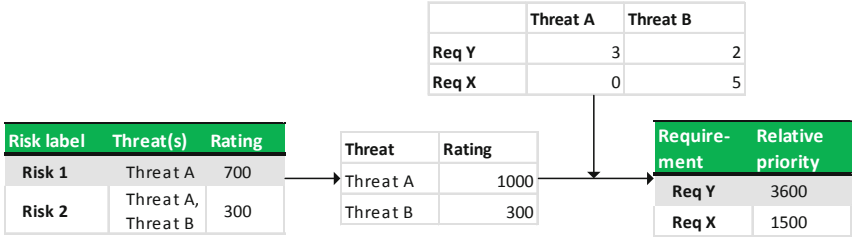

Fig. 3. A simple example of using the threat-requirements map for prioritization

Figure 3 shows a simple example of using the mapping to automatically derive a list of prioritized requirements:

1. Sum up the rating of each risk where a particular threat is mentioned, we obtain a ranked list of threats with relative ratings.
2. Then, for each security requirement, sum up the relative rating of each threat it mitigates multiplied by the threats' relevance factor.

What we end up with is a ranked list of security requirements with relative ratings. These rating is finally normalized to a scale which matches the one used in sprint planning, usually 1 to 4.

5.3 The Security Requirements Repository

An important aspect of the proposed methodology is that it helps the development team account for security requirements during sprint planning by making prioritized security requirements available on the product backlog. Therefore, the ranked list of requirements produced by applying the threat-requirement mapping to the results of the risk assessment needs to find their way into the product backlog. Furthermore, these requirements need to be operationalized. To facilitate this, we propose creating a repository of security requirements in the agile issue tracking software being used by the development team. To be

able to do so, we overload established agile terminology to accommodate security requirements, as shown in Fig. 4 below. The security features stored in the mapping of Sect. 5.2 are defined as Features and their respective requirements are stored as User Stories.

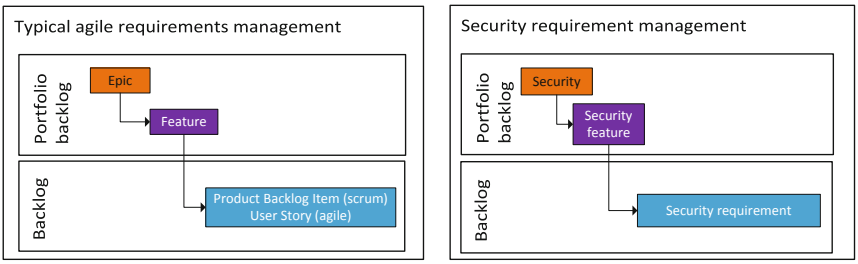

Fig. 4. Casting security requirements into the agile taxonomy

In order to promote accessibility and re-usability of this knowledge, we propose storing the repository as a template project in the software development project management toolkit of choice (e.g. Microsoft TFS/VSTS or JIRA). Based on the results of applying the mapping described in Sect. 5.2, the PO can import the relevant security features from the repository and assign priorities to them. This can be done manually or by means of an extension such as "Issue templates for JIRA"[1]. As long as the selected features are imported together with their children (i.e. the associated backlog items), then all relevant information will be visible on the developer's backlog.

Implementation of the requirements can then take place as per the agile philosophy: the requirements are broken down into tasks, effort estimations are performed, and the tasks are assigned to sprints, based on the priorities of their parent features.

6 First Evaluation

In order to evaluate the proposed framework, we tested each of its three components individually in practical settings. Specifically, we investigated whether:

1. the *risk assessment methodology* is usable;
2. the *threat-requirement map* produces a correct ranking of security features;
3. and the *security requirements repository* is able to store a security requirements knowledge base.

We used ENISA's Threat Landscape [7] as a source for threats and CIP Overheid's Grip on Secure Software Development [13] as a source of security features

[1] https://marketplace.atlassian.com/apps/1211044/issue-templates-for-jira.

and requirements. Both knowledge bases are well established in academia as well as practice and are actively maintained. The threats were added as a drop-down to the risk assessment spreadsheet, the requirements were added to the security requirements repository, and they were both mapped to each other in the threat-feature map.

The Risk Assessment Methodology. We performed a total of five assessments together with the PO and one other stakeholder of five different applications from a variety of domains: finance, HR, retail, social, and privacy. The risk assessment sessions lasted between two and three hours and resulted in the identification of an average of 18 risk scenarios per assessment. All assessments were facilitated by at least one of the authors.

We observed the participants found the exercise engaging and simply going through each threat helped them identify risks they had not considered. We also administered a questionnaire after each session, the results of which are shown in Table 1.

Table 1. Practitioner feedback on the risk assessment methodology

Question	Average rating (1–5)
Has the assessment helped you identify risks?	3.2
Has the assessment helped you understand risks?	3
Has the assessment helped you select security requirements?	3
Would you execute the assessment with clients?	3.2
How easy was performing the assessment?	3.4
Would you recommend the assessment to others?	4.4
Is the assessment suitable for agile processes?	1.8

After the assessment, participants felt they have a better awareness and understanding of the risks their application is exposed to. Even though the questionnaire was administered before the participants were shown the resulting feature prioritization, many stated the assessment helped them think of important security requirements. On average, the participants felt the assessment was not difficult, despite lacking security expertise and having no security experts in the session. Participants also indicated they would perform the assessment with a client and that they highly recommend other teams perform one. The assessment was not deemed suitable for agile processes. However, the assessment is meant as an entry point in order to obtain a list of requirements and priorities without security knowledge; once the requirements are copied to the project backlog, their implementation can take place in an agile way. Furthermore, changes to the application's risk profile can be reflected in the assessment in order to re-calibrate the priorities at any time.

The Threat-Feature Map. In order to assess the feasibility of defining a mapping of threats to security features, the authors manually mapped the 15 threats maintained by ENISA [7] to the 27 security features part of the CIP Overheid's Grip on SSD [13]. For each threat, we evaluated the likelihood (high-medium-low) it would exploit common one of the web-application vulnerabilities used internally for penetration testing. Then, for each vulnerability, we specified which security feature is able to mitigate it. The result was a matrix of relevance factors for each threat-feature tuple.

We asked three security experts in our partner organization to manually assign priorities of 0 (do not implement) to 4 (critical) to each of the security features in our mapping given the results of one of the risk assessment. Each expert was given the assessment of a different application but none of the experts were familiar with the application itself nor were they involved in creating the mapping. The threat-feature map only uses the threat and risk rating columns, however, the human assessors could base their judgment on the entire table.

Table 2 compares the automated prioritization to the manual one for each application. In two of the three assessments the automatically generated results differed significantly from the expert judgment. However, the automatically assigned priorities were similar to the manual ones for the HR system. Across the three assessments 1-in-4 of security features were assigned the same priority by the mapping and the expert. Of the features which were assigned different priorities, 57% deviated by one.

Table 2. Statistical comparison of the automated prioritisation vs. expert judgment

Application	Privacy	HR	Taxes
Correct guesses	3	8	9
Off by 1	14	14	7
Correlation	−0.18	0.28	0.12
p-value	0.82	0.07	0.26

Despite promising results with the HR application, the mapping has overall failed to deliver a prioritization significantly better than random when compared to expert judgment. Either (1) the mapping is incorrect or incomplete, or (2) judging the priorities is difficult and error-prone. Both explanations could be investigated given higher availability of experts by (1) using the expert judgments to infer a mapping or (2) measuring inter-expert agreement.

The Security Requirements Repository. To validate our claim that the structure proposed in Fig. 4 is able to encode any security requirement we defined a template project in Microsoft Team Foundation Server (TFS) and used it to store the entire set of high-level security requirements, operationalized requirements, and control points mandated by the CIP Overheid's Grip on Secure

Software Development. We also wrote a script which is able to import these requirements into any other TFS project. We are currently working on a graphical TFS extension to make this process easier and allow users to also assign priorities during the import (more on this in Sect. 7).

7 Conclusions

Simply performing a risk assessment as described in Sect. 5.1 raises awareness of security issues. Experts did not always agree with the priorities assigned by the threat-requirement mapping, but found manual prioritization difficult. Finally, we showed how the availability, usability, and maintainability of a security baseline can be improved by storing it as a linked collection of backlog items. We believe our framework can help agile development teams take security into account during by providing a first indication of the most important security features, their priority, and the tasks required. Security experts should still be involved during implementation and testing; they can use the assessment and initial prioritization as a starting point or reference.

The proposed framework was developed and tested at a single Dutch software developer, and only applied to mobile and web applications. However, the developer makes use of standard stacks, development practices, and supporting tools. Therefore, following Wieringa [20], we think that the framework could potentially be applicable to other organizations that have similar organizational and software development context to the one of our partnering Dutch company.

Nevertheless, we are looking for industry partners to refine the threat-feature map and strengthen our evaluation. We also want to extend the security requirements repository with technology-specific and domain-specific requirements to enable selection as well as prioritization. Finally, we are exploring using Artificial Intelligence to prioritize requirements based on prioritizations of experts.

References

1. Hammoudeh, A.: A risk-driven approach to security, from check boxes to risk management frameworks (2016). https://securityintelligence.com/a-risk-driven-approach-to-security-from-check-boxes-to-risk-management-frameworks/
2. Baskerville, R.: Agile security for information warfare: a call for research. In: ECIS 2004 Proceedings p. 13 (2004)
3. Beck, K., et al.: Manifesto for Agile Software Development (2001)
4. Boehm, B.W.: A spiral model of software development and enhancement. Computer **21**(5), 61–72 (1988)
5. Daneva, M., Wang, C.: Security requirements engineering in the agile era: How does it work in practice? In: 2018 IEEE 1st International Workshop on Quality Requirements in Agile Projects (QuaRAP), pp. 10–13, August 2018. https://doi.org/10.1109/QuaRAP.2018.00008
6. Davis, A.: Return on security investment-proving it's worth it. Netw. Secur. **2005**(11), 8–10 (2005)

7. ENISA Threat Landscape 2017: 15 Top Cyber-Threats and Trends. Technical report, European Union Agency for Network and Information Security (2017). https://doi.org/10.2824/967192
8. Goldfarb, J.: Risk-driven security: The approach to keep pace with advanced threats (2015). https://www.securityweek.com/risk-driven-security-approach-keep-pace-advanced-threats
9. Information technology - Security techniques - Information security management systems - Requirements. Standard ISO 27001:2005, International Organization for Standardization (ISO) (2005)
10. Information technology - Security techniques - Code of practice for information security management. Standard ISO 27002:2005, International Organization for Standardization (ISO) (2005)
11. Information technology - Security techniques - Information security risk management. Standard ISO 27005:2011, International Organization for Standardization (ISO) (2011)
12. Koers, M., Paans, R., van der Veer, R., Kok, C., Breeman, J.: Grip on secure software development (SSD): 'the client at the helm', version 2.0. Technical report, Centrum voor Informatiebeveiliging en Privacybescherming (CIP), March 2015. https://www.cip-overheid.nl/wp-content/uploads/2018/01/20160622_Grip_on_SSD_The_method_v2_0_EN.pdf
13. Koers, M., Tewarie, W.: Grip on secure software development (SSD): security requirements for (web) applications, version 2.0. Technical report, Centrum voor Informatiebeveiliging en Privacybescherming (CIP), October 2014. https://www.cip-overheid.nl/wp-content/uploads/2018/08/20180821-Grip-on-SSD-Security-requirements-v2.0-2.pdf
14. McDermott, J.: Abuse-case-based assurance arguments. In: Proceedings 17th Annual Computer Security Applications Conference, ACSAC 2001, pp. 366–374. IEEE (2001)
15. McDermott, J., Fox, C.: Using abuse case models for security requirements analysis. In: Proceedings of the 15th Annual Computer Security Applications Conference. (ACSAC 1999), pp. 55–64. IEEE (1999)
16. OWASP: Top 10–2013: The ten most critical web application security risks. The Open Web Application Security Project (2013)
17. Rosenquist, M.: Prioritizing information security risks with threat agent risk assessment. Intel Corporation White Paper (2009)
18. Siponen, M., Baskerville, R., Kuivalainen, T.: Integrating security into agile development methods. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences. HICSS 2005, pp. 185a–185a. IEEE (2005)
19. Terpstra, E., Daneva, M., Wang, C.: Agile practitioners' understanding of security requirements: insights from a grounded theory analysis. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), pp. 439–442. IEEE (2017)
20. Wieringa, R.J.: Design Science Methodology for Information Systems and Software Engineering. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43839-8