The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects

Inger Anne Tøndel
Department of Computer Science, Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
inger.anne.tondel@ntnu.no

Daniela Soares Cruzes
Martin Gilje Jaatun
Kalle Rindell
SINTEF Digital
Trondheim, Norway

ABSTRACT

To achieve a level of security that is just right, software development projects need to strike a balance between security and cost. This necessitates making such decisions as to what security activities to perform in development and which security requirements should be given priority. Current evidence indicates that in many agile development projects, software security is dealt with in a more or less "accidental" way based on individuals' security awareness and interest. This approach is unlikely to lead to an optimal security level for the product. This paper suggests *Security Intention Recap Meetings* as a recurring organisational tool for evaluating current practices regarding the security intentions of a software project, and to make decisions on how to move forward. These meetings involve key decision makers in the project, such as the product owner and the project manager, with the purpose of making security decisions visible and deliberate and to monitor their results.

CCS CONCEPTS

• **Security and privacy → Software security engineering**; • **Software and its engineering → Agile software development**; *Requirements analysis*;

ACM Reference Format:
Inger Anne Tøndel, Daniela Soares Cruzes, Martin Gilje Jaatun, and Kalle Rindell. 2019. The Security Intention Meeting Series as a way to increase visibility of software security decisions in agile development projects . In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES 2019) (ARES '19), August 26–29, 2019, Canterbury, United Kingdom.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3339252.3340337

1 INTRODUCTION

In agile development projects, requirement management is dynamic. As a rule, a development project will not be able to deliver a perfect product within the cost and time constraints [15]. This makes requirements negotiation a key activity. In such an ecosphere, the security work needs to compete for its share of effort and money. Achieving cost-effective security, however, is not an easy task: Understanding and assessing the security needs of the software being under development is challenging in and of itself, further complicated by the complex and constantly changing threat landscape. Without a clear approach to identifying security needs and making decisions on how to address them, software projects are unlikely to end up with cost-effective security.

Studies have found evidence that software security is often dealt with in an "accidental" way in agile projects [23]. It has even been pointed out that *"[a]gile techniques are vulnerable for forgetting things like security."* [22]. Security and quality aspects have a tendency to be sacrificed in favour of implementing more functionality [2, 22, 23], and the decisions involved are commonly made without involving security expertise [23]. The responsibility for software security is often unclear [22, 23] in projects and organisations. Thus, security is not a strategic decision, but rather left up to the individuals involved and their security posture. In particular, the Product Owner has been identified in studies as a common hindrance for sufficiently prioritizing security and quality [2, 22].

Security needs to be considered from the start and throughout a software project, and be visible as an important concern. This is acknowledged in various software security approaches [9, 17] and is reasserted by recent regulation regarding the handling of personal data [7]. Security decisions include decisions on which security activities and practices to perform and what security functionality to implement, but also include other decisions (e.g. design choices) that may have an impact on the security of the produced software. Security decisions are not only made in the beginning of the project, or only at some clearly identified gates, but happen throughout development in big and small ways, sometimes without security being explicitly taken into account.

This paper suggests an approach to bring security priorities and decisions forward in an agile development projects: the *Security Intention Recap Meeting*. This approach is based on ongoing interactions with several development companies [6] and on studies of other security techniques placed in an agile setting, in particular threat modeling [21] [10] and the Protection Poker risk estimation game [26][24]. The security intention recap meeting approach is made for the context of agile software development and project management. These meetings help addressing software security in a systematic way by involving the key decision makers of the project in regular assessments of the current state of the security work, and by comparing how the work is in line with the security intentions for the project. The need to find a balance between security and cost is fully appreciated, and it is advocated that such a balance is unlikely to be achieved without intentional discussions about the right balance for this particular project.

This paper is organised as follows: Section 2 gives an overview of challenges identified in literature on prioritization of software security in agile software development projects. Section 3 presents the concept of the security intention recap meetings and how to organise them into a meeting series. Section 4 discusses the security intention meeting series in relation to other software security activities that are commonly recommended and can have similar goals; that is, threat modeling and risk assessments. Additionally the section identifies and discusses envisioned challenges to applying the approach in practice, explores ways to meet the challenges, and describes plans for future research. Section 5 concludes the paper.

2 CHALLENGES IN PRIORITIZING SECURITY IN AGILE SOFTWARE DEVELOPMENT

In this section, an overview of related studies is given, providing an empirical and theoretical basis for understanding the challenges of getting security prioritised in agile software development. The studies are generally in agreement that security is often neglected or underprioritized [2, 18, 22, 23] and point out many factors and challenges impacting how the security priorities are set in the agile development. The following factors are recurrent in various forms:

- The *individuals* have a key role and their attitude, knowledge, and priorities shape the priorities security is given in the development project. The product owner in particular influence priorities [2, 22], but there are also other important roles (security experts, developers and management at various levels) [13, 14] and there can be tensions between different groups [22, 23].
- The *ownership and responsibilities* for software security are currently unclear [22, 23]. This appears to have a negative impact on the priority given to security [23].
- The *business case* for security is unclear, and the security work is considered a fight not worth fighting [22]. The push for functionality is strong, and this results in less focus on security [2, 22, 23].

In a review of 44 primary studies, Kanniah and Mahrin [13] identified commonly cited factors impacting the successful implementation of secure software development practices. The broad set of factors identified include the institutional context, the people involved and their actions, the project content and the system development process. In a follow-up study with eight experts, the following factors were identified by Kanniah and Mahrin as the ten most important ones: 1) security experts, 2) security documentation, 3) project management, 4) developers, 5) project team, 6) security audit team, 7) team collaboration, 8) development time, 9) policy enforcement and 10) top management [14].

In a literature review of quality requirements work in agile development, Alsaquaf et al. [2] identified the product owner as a hindrance for quality requirements being properly addressed. The product owners commonly have a *"heavy workload"* and *"insufficient availability"*, in addition to a *"lack of knowledge"* of quality aspects [2]. Other challenges include inadequate or lacking techniques, and challenges that functionality is prioritized while some other types of requirements are ignored or insufficiently analysed.

In their systematic literature review, Oueslati et al. identified 14 challenges of developing secure software within the agile approach [18]. The challenges were categorised the following way:

- *"Software development life-cycle challenges"*: security activities not included; hard to integrate security in every iteration due to short iteration times.
- *"Incremental development challenges"*: dealing with changes.
- *"Security assurance challenges"*: documentation; testing; unstable development process.
- *"Awareness and collaboration challenges"*: security requirements neglected; lack of experience and security awareness; separate the developer and reviewer roles.
- *"Security management challenges"*: giving priority to security.

Tøndel et al. [23] studied software security practices among 23 public organisations, using interviews as the main instrument of data collection. This study aimed to identify risk-centric software security practices in organisations and projects in the public domain. It involved people in development and information security positions in organisations mainly using some type of agile software development practices. The findings show that software security work in these organisations is not generally based on security risk, but rather triggered by the requirements for legal and regulatory compliance, or more or less "accidental" detection of security mistakes in development. Barriers against security include unclear responsibilities for software security, architects without an interest in security, lack of security knowledge both on the developer and procurer side, and security being considered a "technical issue". In the organisations studied, it was found that *"[n]o one fights for software security"*, that risk treatment decisions were often *"[a]rbitrary, late and error driven"* and that *"[t]ime pressure results in security requirements being postponed (or even dropped)"* [23].

Terpstra et al. [22] studied practitioners' postings on social media (LinkedIn) to discover how agile practitioners reason about security requirements, and how they cope with them. The analysis resulted in the identification of 21 concepts that indicate problems regarding security requirements in agile, and 15 coping strategies. Problems identified include the limited business value of security, the tendency that security gets lost in the process, and the lack of awareness and knowledge. Their analysis resulted in a descriptive conceptual framework that included the following categories:

- *"[O]wnership of security requirements":* represents the finding that no role assumes, or is given, full responsibility for security requirements in development projects.
- *"[D]efinition of Done (DoD)":* represents the opinion of some professionals that the DoD should represent requirements on the need to implement security measures.
- *"[B]usiness case"*: represents the findings pointing to security not being part of the project's business case.
- *"[A]ttitude towards security requirements":* represents the findings that in some cases *"team members 'do not care' about security requirements just because there is no incentive to do so (...). Or, because no one really understands completely what these requirements are"*.

- *"[O]rganizational setup":* represents the findings that show that the organisational culture can both help and hurt the security requirements work. In particular approaches to educating developers on software security could have an impact.
- *"[P]erceptions of priority":* represents issues related to prioritisation of security requirements at inter-iteration time. Business representatives often drive priorities, pushing for functionality, but their priorities can differ from developers.

3 THE SECURITY INTENTION MEETING SERIES

A security intention (SI) recap meeting is a meeting primarily for decision makers, and is intended to be part of a series where the meetings build on each other. Both these aspects of the meeting are necessary to reach the meeting goal of making software security decisions more visible, systematic and deliberate. In the following we explain how the SI recap meetings are organised into a series and integrated into development before we move on to explaining the different parts of the SI recap meeting in more detail

3.1 Integration into development

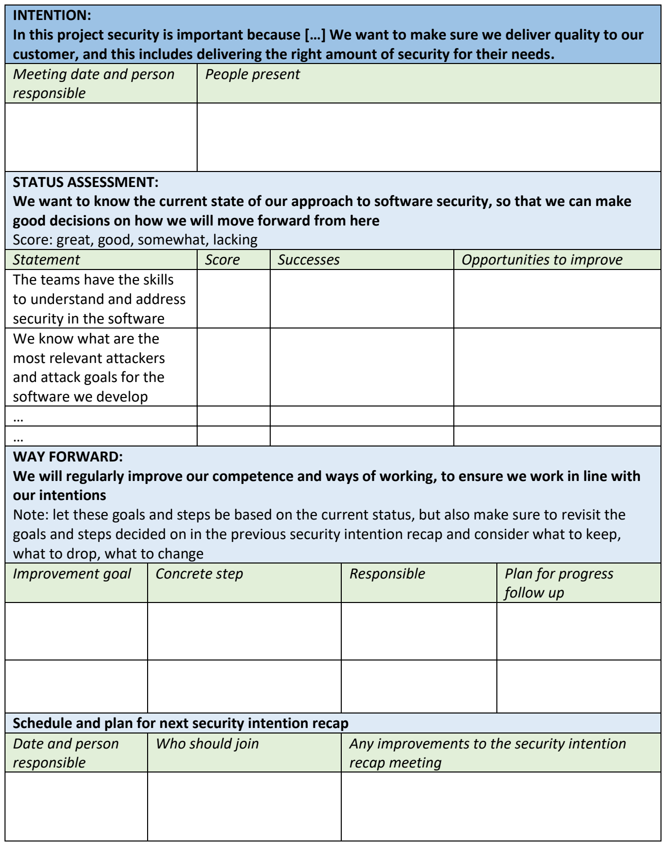
How often SI recap meetings should be held would depend on the product and may vary throughout the project life cycle. Note, however, that the SI recap meetings are meant to be relatively short meetings (ideally maximum one hour), and we advocate to rather have short meetings more often than longer meetings more seldom. Figure 1 gives one example of possible timing in relation to the software development activities of the project.

We envision one initial SI meeting in the beginning of the project, one SI postmortem meeting in the end, and several SI recap meetings during the course of the project. In the initial SI meeting at the beginning of the project, the goal of the meeting is to clarify the overall goals of the software security work in this project, decide on which statements to use for self-evaluation during the project, and decide on initial security activities needed in the initiation of the project. The goal of the SI postmortem meeting is to evaluate the software security approach in this project, related to the goal, and identify learning points for future development projects. The SI postmortem meeting could utilize any postmortem technique [4] and could be part of a larger postmortem meeting for the project, covering more issues than software security.

The SI meeting series, in addition to making software security decisions more visible and deliberate, offers a possibility to document important assumptions, priorities and decisions regarding software security throughout the project. Thus notes should be taken from the meetings and stored as part of the project documentation. Action points from the meeting should find their way into any tools used for issue tracking. The self-evaluation results additionally offer a way to track progress throughout the project and learn more about what types of actions create the effects sought after in order to meet software security goals.

3.2 The SI recap meeting

The SI recap meeting consists of two main parts: 1) an honest evaluation of the current state of software security in the development of the product, and 2) deciding on action points on how to move from here. The meeting shall not be longer than one hour. Figure 2 gives an example template for an SI recap meeting. In the following we explain the key parts of the meeting.



INTENTION:
In this project security is important because [...] We want to make sure we deliver quality to our customer, and this includes delivering the right amount of security for their needs.

Meeting date and person responsible	*People present*

STATUS ASSESSMENT:
We want to know the current state of our approach to software security, so that we can make good decisions on how we will move forward from here
Score: great, good, somewhat, lacking

Statement	*Score*	*Successes*	*Opportunities to improve*
The teams have the skills to understand and address security in the software			
We know what are the most relevant attackers and attack goals for the software we develop			
…			
…			

WAY FORWARD:
We will regularly improve our competence and ways of working, to ensure we work in line with our intentions
Note: let these goals and steps be based on the current status, but also make sure to revisit the goals and steps decided on in the previous security intention recap and consider what to keep, what to drop, what to change

Improvement goal	*Concrete step*	*Responsible*	*Plan for progress follow up*

Schedule and plan for next security intention recap

Date and person responsible	*Who should join*	*Any improvements to the security intention recap meeting*

Figure 2: Template for the security intention recap meeting

3.2.1 *Owner and Participants.* As the SI recap meeting is a meeting that is about making conscious decisions on software security, it is of paramount importance that roles involved in making decisions related to the particular product under development participates in the meeting. Roles such as product owner and project manager should be part of the meeting. Additionally, roles with security responsibility, or responsibility for legal compliance (if relevant) should be part of the meeting. In addition to these roles, it is necessary to have meeting participants that are in touch with what is happening in the actual development. Thus, it may be decided to include one or more developers or testers in the meeting, in particular people with a security champion or a team leader role.

One person needs to be responsible for the SI recap meeting, and for keeping the SI meeting series alive. This person needs to be motivated about software security. Ideally this person should be part of the development project, so that the meeting is not seen as initiated from outside the project.

3.2.2 *Status assessment.* The heart of the SI recap meeting is an honest evaluation of the current state of software security in the development of this particular product. Doing such a self-evaluation

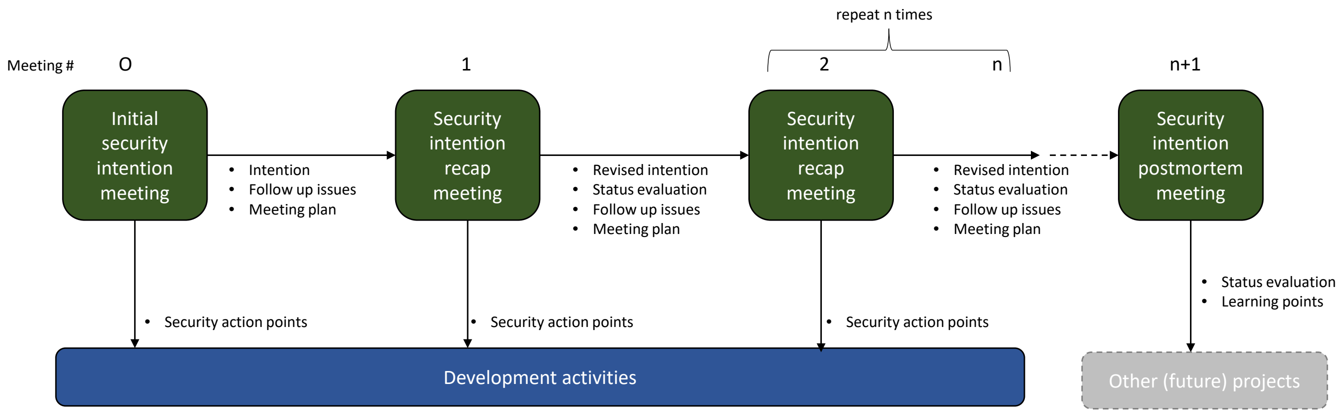


Figure 1: Illustration of the relationship between the security intention meetings and their output

serves two purposes: 1) to remind the participants of the security goals of the project and what level of software security that is aimed for, and 2) to identify areas where there is a need to adjust practices to be more in line with the software security intentions for the product. Current practice is evaluated related to a set of previously selected statements (see subsection 3.2.3) that concretise what level of software security is aimed for in this product For each statement the participants discuss successes and opportunities to improve (inspired by a tool for assessing onboarding [3]). Additionally, we recommend that the meeting participants evaluate each of the statements according to the following scale, to be clear about where the practices are acceptable and where improvement is needed:

- *Great:* We are doing great, and do not need to prioritize further improvement in this area
- *Good:* We know we could be better, but are fairly satisfied with current practice
- *Somewhat:* We are doing some things, but really should improve this part
- *Lacking:* We are doing close to nothing and are far from realizing this goal

Discussing and documenting both an evaluation of current practice and the basis for this evaluation (successes; opportunities to improve) is the foundation for making decisions on the way forward.

In the self-evaluation part of the meeting it is essential that all participants have their say so that the self-evaluation ends up being as true as possible related to the current state. One wants to avoid one or two meeting participants dominating the evaluation, leaving out other perspectives. It is possible to use a voting mechanism similar to that used in Planning Poker [8] or Protection Poker[25, 26] to ensure all participants make an individual assessment of each statement, and that each individual assessment is made visible in the meeting. In any case, the moderator of the SI recap meeting is essential in creating a safe atmosphere for discussion and making sure all relevant voices are heard in the meeting.

3.2.3 Process for selecting statements for self-evaluation. Each project should select a manageable set of statements to assess for the project. Selecting statements for self-evaluation is a way of making priorities for the project, as these statements will be used to guide attention and decisions in the project. Selecting some statements implies not selecting others. As priorities need to be made related to the particular project, we do not provide a finished list to choose from, but rather a process for selecting statements. The self-evaluation statements should be decided on in the beginning of the project, but can be revised as the project moves along if underlying assumptions or overall priorities change related to software security, or if one has reached the goals on one statement and wants to put the focus elsewhere. The template in Figure 2 shows example content for the self-evaluation statements.

Statements can be identified top-down, based on priorities on an organisational level, or bottom-up, based on the individual project, or a combination. We suggest that the first step in deciding on a set of self-evaluation statements is to answer the following questions:

- Does the organisation have any strategies that sets out the goals or ambitions regarding software security? Examples of such documents would be software security manifestos or KPIs.
- Does the organisation know its strong and weak spots when it comes to software security, and have identified areas of improvement? Examples would be results from a BSIMM or OpenSAMM evaluation of practices.
- Does the product being developed have any specific characteristics that can influence software security and make software security different than in most other products we develop? Examples could be customer expectations, legal requirements,technology, and exposure of the software.
- Are there any aspects of the team(s) involved that impact our ability to do software security well? Examples could be security competence and awareness, and team culture.

By answering the above questions one would identify the main sources for the self-evaluation statements. The next step would be to identify possible statements from these main sources. Then the final step would be to choose a manageable set of self-evaluation statements for the project. We suggest to start with 5 to 7 statements. If assuming that five minutes would be enough for an evaluation of each statement, that would imply from 25 to 35 minutes of the meeting spent on status assessment.

Projects need to balance the need for having self-evaluation statements that are something to aim for and give a motivation to improve, and the need for realistic statements. We recommend that the statements selected represent the real ambitions of the project, meaning that if a statement is met then the project is at the right level of security in that area.

To give an idea of how self-evaluation statements can be identified in practice, Table 1 shows potential self-evaluation statements that have been made based on the DevSecOps manifesto inspired from the Build-Security-In Manifesto and the Secure Development Lifecycle initiative principles at Comcast [16], BSIMM scores from an evaluation of software security maturity in public organisations [11] and for applications that handle personal health information that is subject to legal requirements [12].

3.2.4 Way forward. Based on the status assessment, the participants in the SI recap meeting should decide on concrete action points that would move the state of software security more in line with the goals for the product. Note that this may mean to start or improve some software security initiatives (e.g. do a threat modeling session, do a training session on a specific software security topic, do a risk assessment, increase security testing efforts, etc.), but it can also mean to stop or reduce efforts in one or more existing software security activities. The action points decided on need to be concrete and have a deadline and someone responsible in order to increase likelihood that the action points will be followed up in the day-to-day development activities. To increase the commitment to the action points, we would suggest that the SI recap meeting participants, as part of the "way forward" part of the meeting, revisit decisions from the previous meetings to see if the previous action points have been followed up, and if not, discuss how to increase the likelihood that the action points decided upon in the current meeting will have more of an impact.

One important part of deciding on the way forward related to software security is to decide upon when the next SI recap meeting should be held for this product, and who should participate. The reason we suggest that this is decided upon in this meeting, and put into the calendars of the participants, is to reduce the likelihood that the commitment to having these meetings is forgotten.

4 DISCUSSION

This section explains how the SI recap meeting is complementary to other software security techniques such as risk assessment and threat modeling. It moves on to looking at some known challenges related to adoption of threat modeling and a risk estimation technique called Protection Poker. These already identified challenges are then used to describe likely challenges to the SI recap meetings so that these challenges can be proactively addressed. Finally, we describe future research endeavors to evaluate and improve the SI recap meeting approach.

4.1 Relation to other software security techniques

Threat modeling [21] and risk assessment [23] activities can be used to make decisions and priorities on how to move forward based on an assessment of the current status. Compared to an SI recap meeting, the status assessments made in these types of activities are normally on a much lower level of abstraction and with an emphasis on the system and what can go wrong. Thus these activities usually take longer than what is envisioned for an SI recap meeting. Often these activities are performed by participants with technical competence and leave out decision makers. The SI recap meetings are not an alternative to risk assessment or threat modeling, but rather a place where the decision to perform or not perform risk assessment or threat modeling activities could be made.

Protection Poker [25, 26] is a security risk estimation game that is particularly suited for agile teams. It offers a practical way of doing risk assessment in an iterative fashion, and looks at the assets and the ease of attack that comes with implementation of features. Compared to the SI recap meetings, Protection Poker's goal is less geared towards decision making. The participants are different, with Protection Poker involving the entire team. The time it takes to play Protection Poker can vary from team to team, depending on the discussions and their familiarity with the game. However, as Protection Poker is intended to be played for every iteration with the full team, the total time it takes would likely be much longer than an SI recap meeting.

4.2 Potential Challenges

The SI recap meeting has not yet been tried out in practice in development companies. The suggested approach is a response to reported challenges in literature on having security being given the "right" priority in agile software development projects, as well as our own experiences with ongoing interactions with software companies on software security [6]. Though the SI recap meetings are different than techniques such as threat modeling and Protection Poker, we believe it would face some of the same challenges to adoption. Thus we have looked to a study of adoption of Protection Poker [24] and a study on applying Microsoft Threat Modeling to agile projects [5] to identify what we believe are likely challenges to adoption of the SI recap meeting approach. In the following we explain these envisioned challenges and provide suggestions for how to address them. Table 2 gives an overview of how challenges identified for Protection Poker and threat modeling relate to the envisioned challenges to the SI recap meetings.

4.2.1 Perceiving improved software security as a consequence of the SI recap meeting. It is a likely challenge that the SI recap meeting is viewed as "yet another meeting". For both Protection Poker and threat modeling it was challenging to see clearly how the technique led to improved security of the software, and not only discussions about security. The SI recap meeting, is likely to face the challenge of having a visible and traceable direct impact on the delivered security of the code.

To address this challenge, the SI recap meeting needs to ensure that the meeting leads to actionable decisions that are followed up in development. Having participants with the authority to make decisions and have them implemented is thus of key importance. At the same time, it is important that the development team(s) have confidence that the decisions reached in these meetings are good ones. Thus the competence of the participants is important, both related to security and the overall understanding of the product, as

Source	Issue	Potential self-evaluation statements
DevSecOps Manifesto [16]	*"Build security in more than bolt it on"*;*"Implement features securely more than security features"*	• We consider security in the design of all functionality, not only for security features.
	"Rely on empowered engineering teams more than security specialists"; *"Build on culture change more than policy enforcement"*	• The teams have the skills to understand and address security in the software. • The teams feel responsible for how the software behaves in production, including security implications.
	"Use tools as feedback for learning more than end-of-phase stage gates"	• We use security testing tools early to give feedback to developers and improve their skills in writing vulnerability-free software.
BSIMM scores [11]	Attack Models is the area with lowest maturity Strategy and Metrics is the area with the second lowest maturity	• We know what are the most relevant attackers and attack goals for the software we develop. • We have a clear processes for software security, and this process is known and followed by the development team.
Product characteristics	Health information	• We meet legal requirements for protection of health related data

Table 1: Example self-evaluation statements and their sources

Envisioned SI recap meeting challenge	Related Protection Poker (PP) challenges [24]	Related Threat Modeling challenges [5]
Perceiving improved software security as a consequence of the SI recap meeting	• PP did not improve security of the software • Ensuring confidence in the results • Important aspects from the discussion is lost • The output from playing PP is not concrete in terms of what to do next	• Documentation of the assets after the meeting was not done • Many discussions on threats and mitigation strategies get lost • The approach does not make a link to the actual code • It is hard to know when enough analysis has been done • The output of the sessions are a list of concerns/threats that are not concrete • Follow up of the threats is challenging
Running and facilitating the meeting	• It is difficult to reach consensus, something that results in a lot of time spent and sometimes results in tension in the team • Some team members may end up with too much influence	• The meeting needs to be structured, but it is not always clear on how to run the meeting • It is hard to know which other people should be included in the meetings besides the "core" development team • There are challenges with running meetings in distributed settings • The meetings are not effective
Selecting self-evaluation statements	• Starting to use PP is time consuming due to calibration and the need to identify and play about assets • Selecting granularity of assets and assigning value to assets can be challenging	• It is challenging to motivate the teams to draw the diagrams • It was hard to decide on the right level of abstraction to the DFDs • It takes long time to draw the diagrams
Establishing the SI recap meeting as a regular event	• Teams did not end up using PP in a regular fashion • Planning meetings are already full • PP takes too much time	• There is a need for a security expert to run the meeting; not every team has this profession available • It is not easy to have everyone participating

Table 2: Selected challenges from study of Protection Poker[24] and Microsoft Threat Modeling [5]

well as how the decisions made and their rationale is communicated outside of the SI recap meeting.

4.2.2 Running and facilitating the meeting. Having effective security meetings where everybody's opinion is heard and valued, while at the same time aiming to reach some kind of consensus, is challenging. As in the Protection Poker meetings [24], at the SI recap meetings there will likely be some participants with more authority than others, and there is a risk that these may end up influencing other participants to the extent that important perspectives are lost. This risk comes in addition to common challenges on how to run meetings, how to know who should participate and how to deal with distributed settings, as was found for threat modeling [5].

Addressing these challenges fully is difficult, as it involves balancing somewhat conflicting goals (efficiency vs. including many perspectives). However, having a skilled facilitator would be an important step in ensuring quality of the meetings themselves.

4.2.3 Selecting self-evaluation statements. Protection Poker and threat modeling approaches do not use self-evaluation statements as we propose for the SI recap meetings. However, they need other kinds of preparations (for example, calibration and possibly asset identification and evaluation for Protection Poker, Data Flow Diagrams (DFDs) for threat modeling). These preparations require an upfront investment in time and effort. Experiences from Protection Poker and threat modeling show that it can be challenging to motivate participants for these preparatory activities, and that they can be perceived as time consuming. The tasks can additionally be challenging when it comes to "doing them right", e.g. at the right level of abstraction that makes them useful in the upcoming activities.

As has already been pointed out, identifying self-evaluation statements is a challenging task, and one that is important as it guides future priorities. It is likely that this preparatory task will require time and effort from key people if projects are to arrive at an optimal set of self-evaluation statements. It is hard to foresee how this will play out in practice before we start experimenting with it in real companies and software development projects. We expect that the process for selecting self-evaluation statements that is laid out in subsection 3.2.3 will be improved substantially based on future experiences with this part of the approach.

4.2.4 Establishing the SI recap meeting as a regular event. The SI recap meetings are intended to be regular events that will serve as reminders of security commitments and increase visibility of security decisions throughout the development of an application. Making the SI recap meetings into regular events is important to increase visibility and awareness of all the ways security decisions are made throughout the project, and influence these in a more deliberate way. However, establishing a new regular practice is challenging. It takes commitment from the project team over a long period of time. In the studies of Protection Poker and threat modeling, time issues and having people participate was considered challenging. Another study of Protection Poker found that although the technique was found to have important benefits, the team still stopped using Protection Poker some time after the study for unknown reasons [26].

As already pointed out in section 3, it is important that one person is responsible for the SI recap meeting and for keeping the SI meeting series alive, "championing it". Finding such a person, that is both interested enough in software security and has the necessary influence on the development project and ability to motivate others, can however be challenging. Not every project may have such a person available.

The SI meeting series we propose is by design lightweight and can be easily adjusted to the needs of the project and the organisation. Adjustments can be made regarding meeting schedule, participants and self-evaluation statements, something that may ease adoption of the technique. Additionally, it is possible to argue that the meeting has the potential to save effort in the longer run, as making more deliberate security decisions and priorities may lead to reduced costs later on (e.g. through not spending time on more security activities than necessary, and by reducing the need for expensive changes that stem from big "surprises" related to security implications of features or design choices). This however does not take away the need for someone that is willing and able to push for the meeting in a way that actually leads to longer-term adoption.

4.3 Further Work

Though the SI recap meeting approach builds on challenges and needs that have been identified in previous research and in our own continuing collaboration with software security companies [6], the approach itself has not yet been tried out and validated empirically. In the future, we plan to foster the adoption of this approach in some of the companies we collaborate with and then collect experiences and improve the approach. In particular we are interested in investigating the following research questions:

- *Adoption:* How is the approach received in the companies? What makes them interested in adopting the approach, and how can one support long-term adoption?
- *Effects:* How does the SI recap meeting influence the development, and what can be done to increase the positive effects of using this approach while minimising the cost?
- *Support:* How can agile projects be supported in using the SI recap meeting approach? Where is support most needed and what are the recommendations that are most important to give to projects wanting to adopt the approach when it comes to frequency, length of meeting and participants?
- *Self-evaluation statements:* How to select statements in a way that is motivating? How can projects be supported with recommendations of what is important statements for their kind of projects?

Additionally, we would welcome more research that can contribute to understanding what can be done to support longer term adoption of software security activities and that can help understand what are the most important goals and practices to strive for in an agile project when it comes to software security. Better understanding of these aspects can feed the content of the SI recap meeting, but also software security priorities in a more general sense.

5 CONCLUSION

This paper proposes the security intention meeting series as a way to make software security decisions more visible, systematic and deliberate in agile development projects. By tackling current challenges that security is easily "forgotten" in agile development [22] and sacrificed for functionality [2, 22, 23] and that security priorities are highly dependent on the varying interests of the individuals involved [2, 22], projects are more likely to move towards cost-effective software security. In order to achieve this, the SI recap meetings needs to be adopted by software projects and organisations and integrated into their way of working.

ACKNOWLEDGMENT

This work was supported by the *Science of Security in Agile Software Development* project (SoS-Agile), funded by the Research Council of Norway (grant number 247678).

REFERENCES

[1] Icek Ajzen. 1991. The theory of planned behavior. *Organizational Behavior and Human Decision Processes* 50, 2 (1991), 179 – 211. https://doi.org/10.1016/0749-5978(91)90020-T Theories of Cognitive Self-Regulation.

[2] Wasim Alsaqaf, Maya Daneva, and Roel Wieringa. 2017. Quality requirements in large-scale distributed agile projects–a systematic literature review. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 219–234.

[3] Talya N Bauer. 2010. Onboarding new employees: Maximizing success. *SHRM Foundation's Effective Practice Guideline Series* 7 (2010).

[4] Paulo Caroli and Taina Caetano. 2015. *Fun Retrospectives - Activities and ideas for making agile retrospectives more engaging*. Leanpub, Layton.

[5] Daniela Soares Cruzes, Martin Gilje Jaatun, Karin Bernsmed, and Inger Anne Tøndel. 2018. Challenges and Experiences with Applying Microsoft Threat Modeling in Agile Development Projects. In *2018 25th Australasian Software Engineering Conference (ASWEC)*. IEEE, 111–120.

[6] Daniela S. Cruzes, Martin G. Jaatun, and Tosin D. Oyetoyan. 2018. Challenges and Approaches of Performing Canonical Action Research in Software Security: Research Paper. In *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HoTSoS '18)*. ACM, New York, NY, USA, Article 8, 11 pages. https://doi.org/10.1145/3190619.3190634

[7] EU. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *L* 119 (2016).

[8] James Grenning. 2002. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting* 3 (2002), 22–23.

[9] Michael Howard and Steve Lipner. 2006. *The Security Development Lifecycle*. Microsoft Press.

[10] Martin Gilje Jaatun, Karin Bernsmed, Daniela S. Cruzes, and Inger Anne Tøndel. 2019. Threat Modeling in Agile Software Development. In *Exploring Security in Software Architecture and Design*, Michael Felderer and Riccardo Scandariato (Eds.). IGI Global.

[11] Martin Gilje Jaatun, Daniela S. Cruzes, Karin Bernsmed, Inger Anne Tøndel, and Lillian Røstad. 2015. Software Security Maturity in Public Organisations. In *Information Security*, Javier Lopez and Chris J. Mitchell (Eds.). Lecture Notes in Computer Science, Vol. 9290. Springer International Publishing, 120–138.

[12] J. Jensen, I. A. Tøndel, M. G. Jaatun, P. H. Meland, and H. Andresen. 2009. Reusable Security Requirements for Healthcare Applications. In *2009 International Conference on Availability, Reliability and Security*. 380–385. https://doi.org/10.1109/ARES.2009.107

[13] Sri Lakshmi Kanniah and Mohd Naz'ri Mahrin. 2016. A review on factors influencing implementation of secure software development practices. *World Academy of Science, Engineering and Technology, International Journal of Social, Behavioural, Educational, Economic, Business and Industrial Engineering* 10, 8 (2016), 2860–2867.

[14] Sri Lakshmi Kanniah and Mohd Naz'ri Mahrin. 2018. Secure Software Development Practice Adoption Model: A Delphi Study. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 10, 2-8 (2018), 71–75.

[15] Dean Leffingwell. 2010. *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.

[16] Larry Maccherone. 2017. The DevSecOps Manifesto. https://medium.com/continuous-agile/the-devsecops-manifesto-94579e0eb716. (2017). Accessed: 2019-04-30.

[17] Gary McGraw. 2006. *Software Security: Building Security In*. Addison-Wesley.

[18] Hela Oueslati, Mohammad Masudur Rahman, and Lotfi ben Othmane. 2015. Literature review of the challenges of developing secure software using the agile approach. In *10th International Conference on Availability, Reliability and Security (ARES)*. IEEE, 540–547.

[19] James O Prochaska. 2008. Decision making in the transtheoretical model of behavior change. *Medical decision making* 28, 6 (2008), 845–849.

[20] Ronald W Rogers and Steven Prentice-Dunn. 1997. Protection motivation theory. In *Handbook of health behavior research 1: Personal and social determinants*. Plenum Press, 113–132.

[21] Adam Shostack. 2014. *Threat Modeling: Designing for Security*. Wiley.

[22] Evenynke Terpstra, Maya Daneva, and Chong Wang. 2017. Agile Practitioners' Understanding of Security Requirements: Insights from a Grounded Theory Analysis. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, 439–442.

[23] Inger Anne Tøndel, Martin Gilje Jaatun, Daniela Soares Cruzes, and Nils Brede Moe. 2017. Risk Centric Activities in Secure Software Development in Public Organisations. *International Journal of Secure Software Engineering (IJSSE)* 8, 4 (2017), 1–30.

[24] Inger Anne Tøndel, Laurie Williams, Daniela Soares Cruzes, and Martin Gilje Jaatun. 2019. Collaborative Security Risk Estimation in Agile Software Development. *Information and Computer Security* (2019).

[25] Laurie Williams, Michael Gegick, and Andrew Meneely. 2009. Protection poker: Structuring software security risk assessment and knowledge transfer. In *International Symposium on Engineering Secure Software and Systems*. Springer, 122–134.

[26] Laurie Williams, Andrew Meneely, and Grant Shipley. 2010. Protection poker: The new software security game. *IEEE Security and Privacy* 8, 3 (2010), 14–20.