Threat Poker: Solving Security and Privacy Threats in Agile Software Development [⋆]

Hanne Rygge and Audun Jøsang

University of Oslo, Norway
`hanneryg@ifi.uio.no, audun.josang@mn.uio.no`

Abstract. Secure software development is a part of 'security by design' which in turn is a prerequisite for 'privacy by design' defined by GDPR. To follow and adhere to the principles of privacy and security by design during software development is a legal requirement throughout Europe with the introduction of GDPR in 2018. Secure software development is typically based on specific methods that software-design teams apply to discover and solve security threats and thereby to improve the security of systems in general. This paper describes Threat Poker as a team-based method to be exercised during agile software development for assessing both security risk and privacy risk, and for evaluating the effort needed to remove corresponding vulnerabilities in the developed software.

1 Introduction

The adoption and application of practical methods for developing adequately secure software has become a necessity for companies and developers in order to produce legally compliant IT systems. The trend towards increasingly prescriptive security and privacy regulations for IT systems such as GDPR (General Data Protection Regulation) has resulted in very specific requirements with regards to security and privacy in IT systems. Threat modeling and removal of relevant vulnerabilities can be considered as the main elements which contribute to strengthening the security of IT systems.

We introduce Threat Poker as an efficient method for secure systems development which stimulates developers to consider security and privacy threats and to evaluate ways to remove or mitigate vulnerabilities related to those threats. Threat Poker is a card game meant to be played during Scrum meetings or other team meetings in agile software development projects. The idea behind the game is to stimulate the team members to think about – and discuss – relevant threats and risks to security and privacy resulting from each new feature or user story being introduced and implemented in the incrementally completed system.

Threat Poker is in some aspects similar to Planning Poker [5], but while Planning Poker is used mainly for time estimation of implementation efforts, Threat Poker primarily focuses on the estimation of risks resulting from security

[⋆] Published in the proceedings of the 23rd Nordic Conference on Secure IT Systems (NordSec 2018), Oslo, November 2018.

and privacy threats, and secondly on the estimation of the effort needed to solve those threats, i.e. to remove relevant vulnerabilities. As required by GDPR, it is necessary to consider both the security risks as well as the privacy risks during the software development process. A security risk represents the potential severity of a threat to cause harm which negatively affects information assets. A privacy risk on the other hand represents the potential severity of a threat to cause privacy harm to data subjects, i.e. to negatively affect the privacy of real persons affected by the data. The victim of a threat is thus totally different in the case of security risks and privacy risks respectively, which precisely is the reason why these two risk types must be considered separately.

Section 2 below briefly mentions different models for software development, in particular the waterfall and agile models. Section 2.1 offers a description of how security is currently implemented in methods of software development. Section 2.2 describes the details of how to play Threat Poker. Section 3 describes qualitative observations of a simple experiment with students at the University of Oslo participating in a session of Threat Poker as part of a class in agile software development. Section 4 provides a discussion and draws conclusions from this initial study.

2 SDLC - Software Development Lifecycle

Several software development models or approaches have been proposed and applied during the last 30 years. Each model has its characteristics, advantages and disadvantages, but common to them all is that they do not focus on security [6] (p.1098). A selection of five prominent development models are briefly analysed and compared in [8]. These are: *Waterfall, Iteration, V-shaped, Spiral* and *Agile*. Scrum is a particular form of the Agile Model which is discussed below.

The waterfall method is the classical and most heavy-weight approach to software development, whereas agile methods represent the most light-weight and flexible approach. We will briefly describe the waterfall method and the specific agile method called Scrum, as they represent very different approaches to software development. Figure 1 shows the waterfall model.

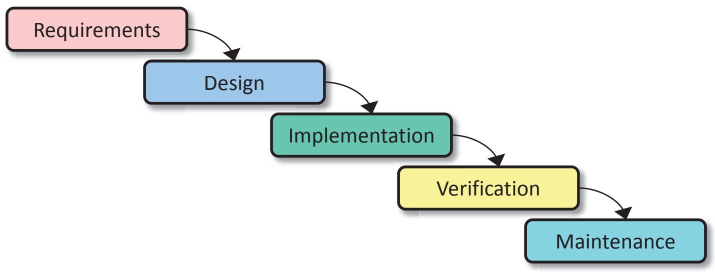

Fig. 1. The waterfall model for software development

The basic idea behind the waterfall model is that each phase must be fully completed before the next phase, as symbolized by the waterfall metaphor where water only flows downwards. This also implies that the complete set of requirements must be defined and fixed at the beginning of the project. In case it is necessary to revisit a previous stage, then a costly overhead is to be expected (metaphorically make water flow upwards), so this should be avoided. However, it is typically the case that requirements have to be changed in the middle of a software development project, so that many software development projects based on the waterfall model have suffered large blow-outs in cost and time.

As a reaction to the rigid structure of the waterfall model several other models have been proposed, where the Scrum method is illustrated in Figure 2 below.

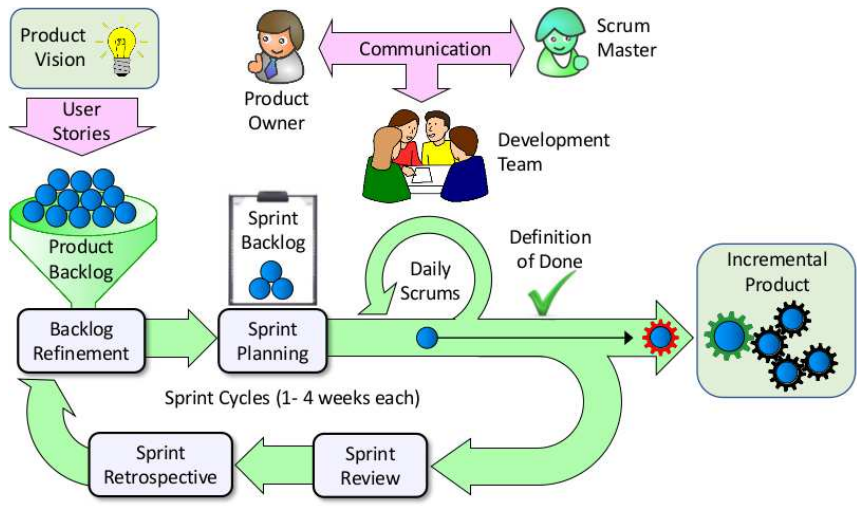

Fig. 2. The Scrum method for software development

The basic idea behind agile methods such as Scrum is that new or evolving requirements can be specified in parallel with, or after already implemented requirements [2]. This is possible by splitting the development into separate *stories* where each story covers a set of requirements that can be implemented and tested more or less independently of other stories. Each cyclic iteration in the agile model is a *sprint* which are to be completed in a specific amount of time, usually a few weeks. The major drawback of the agile model is that it often does not scale well to large and complex development projects.

Specific security related tasks should be included in the various phases of the SDLC, whether the development follows the waterfall model, the agile model, or any other model. Due to the radical difference between the waterfall and agile models, the development team needs to adapt the specific approach to secure development depending on the model followed, as described in the next section.

2.1 Secure Software Development

Security Design Principles. When considering security by design, i.e. to adequately consider security during every step of the development process, it can be helpful to consider common design principles. Several principles are defined by OWASP (Open Web Application Security Project) in ASVS (Application Security Verification Standard) [9]. These principles consists of: minimizing the attack surface, establishing secure defaults, the principles of least privilege, defence in depth, fail securely, don't trust services, separation of duties, avoid security by obscurity, keep security simple and fix security issues correctly. Security principles are worth considering when working on secure development, and they often serve as a guide to what good design could look like. These are also helpful for developers who might not have that much experience with security because it gives an idea of what they should look for.

Secure Software Development in the Waterfall Model. Several models have been proposed to ensure secure development in the waterfall model, including the NIST framework for Security Considerations in the System Development Life Cycle [7], as well as Microsoft's Security Development Lifecycle (SDL) [3].

In Microsoft SDL, every phase of the waterfall model includes security related tasks, which thereby contributes to reducing the number of vulnerabilities in the delivered software.

Vulnerabilities are unintentional side-effect of software development. The challenge for secure software development is precisely to avoid building vulnerabilities into the system. By looking at the list of 25 most common software errors maintained by SANS[1], we see that many of these are directly related to irresponsible or sloppy programming practice.

Secure Software Development in the Agile Model. Agile development can follow different development methodologies that all share a common set of values and principles. These principles include allowing continuous changes in the specifications, frequent deliveries, frequent meetings, both between the developers and business people, but also internally in the development team. Progress is measured by working software, and the process should promote sustainability. Simplicity is essential, as well as technical excellence and good design. The teams should be self-organizing and at regular intervals, should reflect on the process and adjust accordingly [2]. Agile methodologies are based on the continuity of the many different processes involved, the planning, the testing, integration and others. All agile methodologies are made to be lightweight and stress the importance of collaboration between team members and encourage them to quickly and efficiently reach a decision. [16] [11]

There are relatively few studies in the literature on secure agile software development models. In Wichers' proposal [17] it is argued that secure software

[1] http://www.sans.org/top25-software-errors/

development in the agile model needs a quite different approach to that of the waterfall model.

In [17] it is recommended to identify all stakeholders and clarify what their main security concerns are. From that analysis a set of threat models can be extracted which in turn form the basis for stakeholder security stories. Then during the development phase, one has periodic security sprints in between the regular development sprints. It is also proposed to include a final security review before deploying the final system.

Microsoft has presented a version of SDL for agile software development [4]. The Agile SDL model contains the same security steps as in the waterfall SDL model, where these steps are grouped in 3 categories:

– One-Time practices: Foundational security practices that must be established once at the start of every new Agile project.
– Every-Sprint practices: Essential security tasks done in every sprint.
– Bucket practices: Important security tasks that must be done regularly, but not necessarily in every sprint.

Microsoft's agile SDL model has merit, but one drawback is that it does not separate between functional and non-functional security requirements. The model for secure agile software development the we propose below is partially inspired by the model described in [17] and by Microsoft's Agile SDL model, and at the same time offers several improvements over the models mentioned above. Our approach to handling security in the agile model is based on the distinction between functional security controls and non-functional security controls, as described below.

– **Functional security controls** reflect and implement user stories that are directly related to security, such as when password management and verification is used as a control to implement a user story for logon, or when ACLs (Access Control Lists) are used as a control for specifying and enforcing policies for accessing various resources within a domain.
– **Non-functional security controls** are applied in order to eliminate or mitigate vulnerabilities in the implementation of other user stories, such as when applying secure programming techniques in order to avoid buffer overflow bugs, or when applying input filtering when designing a front-end to an SQL database in order to avoid SQL-injection. Software designers must understand that any type of user stories, both ordinary user stories as well as specific security related user stories, must be implemented in a secure way. The way to do that is precisely through non-functional security controls. The idea is that security threats that are intrinsic to a specific user story should be handled during the sprint for the same user story.

A further example of non-functional security controls is when implementing a user story about the logic for handling the check-out of a shopping basket on an e-commerce website, where a threat could be that the customer is able to trick the system into changing the number of items after the price has been computed, so that the customer could receive many items but only pay for one. This security concern must be handled during the sprint that implements the check-out of shopping baskets. Based on these considerations we propose to introduce a new security phase into the sprint iteration. This security phase focuses specifically on identifying threats against the current user stories. The new phase should also specify how the threats can be controlled or mitigated, and should specify tests for those mitigation controls. The implementation of non-functional security controls is then handled in the ordinary phase that develops, integrates and tests the new functionality for the current sprint. Threat Poker is precisely the technique to be used in the security phase.

The Scrum Method. Scrum is defined as "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value." [12] It was introduced in the 1990's by creators Ken Schwaber and Jeff Sutherland, and has been used to great effect by helping to improve the product as well as the teams working on them. The Scrum method is built around Scrum teams and their predefined roles as well as certain rules and events, and each component is an integral part of the scrum process and is essential to the method. Each scrum team consists of members that will have different roles with different functions. These roles consists of a Product Owner, the development team, and the Scrum Master.

The scrum team needs to adhere to certain events defined in the scrum method. These include the sprint, the sprint planning, the daily scrum, the sprint review and the sprint retrospective. The Scrum method also consists of a *product backlog*, which is everything that is needed in the product, and the *sprint backlog*, which is a subset of the product backlog, certain items that are to be completed in the current sprint and the plan for how this increment of the product is to be delivered.

Security Backlog in Scrum. Like many development models, Scrum is not built around implementation of security and does not provide a guide on how to deal with the security aspects of the software that is being developed. Due to this, there needs to be an effective way to implement security features into Scrum and other agile methodologies. A way to do this is with the Security Backlog. The Security backlog is a new backlog that is added to the scrum model, and is used to manage the security risks associated with the software. In addition to the backlog, another role is proposed to add to the Scrum model, the Security Master, who should be a person who has considerable knowledge about security and has the job of managing the Security Backlog. The Security Backlog implements the security design principles to limit the vulnerabilities and to reduce the security risks of the software. Using this method, forces the Product Backlog to go through the Security Backlog and the Security Master decides which features require security attention and these are added to the sprint backlog to be carried out by the developers. Other security features that the Security Master selected are added to the the Security Backlog. [1]

Secure Scrum. Another method used to implement security in scrum is the Secure Scrum Model. Secure Scrum consists of four different components, Identification component, Implementation component, Verification component, and Definition of Done component. These components are put on top of scrum and they are used to influence the stages of the scrum process. In the implementation component, security concerns are identified and marked in the Product Backlog. The implementation component raises the awareness of the security concerns which is used in Sprint Planning and the Daily Scrum meetings. The verification component ensures that testing is possible with focus on security. Last, the Definition of Done defines the Definition of Done for security related issues. [10]

Protection Poker. *Protection Poker* is a method that can be used by developers to conduct software risk assessment when working with agile development methods [13]. It was proposed by professor Laurie Williams at North Carolina State University [18]. It was developed as a means to help agile development teams prioritize security so as to prevent the attacks that could cause the most damage. Protection Poker is a security game played with playing cards. It is made to be played during planning meeting, if using scrum, it would happen in the scrum planning sessions. During these sessions, the product owner will explain the requirements of the feature to be developed. When the requirements are understood, the process progress to a new discussion in the team where they discuss the security ramifications this new feature could have. Misuse cases and threat models must be examined to determine how the new feature may impact the system, if it will make the system more or less secure, or whether the security is impacted at all, and talk of how this might be solved. When this part of the process is completed the participants will vote using the playing cards on the security risk components according to the traditional security estimation model

$$\text{Risk} = (\text{likelihood of incident}) \times (\text{impact of incident}). \tag{1}$$

The team players will express their estimation of the risk (likelihood of an incident and its impact) by throwing cards face down, and then discuss their estimations when the card values are revealed. This process can continue over repeated rounds until a consensus is reached.

Protection Poker is a *Wideband-Delphi technique*, which is based on the Delphi practice which was developed in the 1940s by the RAND Corporation and is used for making forecasts. In this practice participants make individual and anonymous estimations, and show the result, but does not discuss the thoughts behind them. Wideband Delphi was created as a variant to the original where discussions would occur between each repeated round [18].

Elevation of Privilege. Microsofts Elevation of Privilege is also an existing method used to help with threat modeling. It was originally intended for those new to threat modeling as well as those who do it occasionally, to expose them to threat modeling and with it being a game, to bring some enjoyment to threat modeling for non-security experts. The Elevation of Privilege game is played using a special deck of cards that consists of 84 cards, 74 of which are playing cards and the rest divided into instruction cards, reference cards, an about card, and a play and strategy flowchart card. The 74 playing cards are divided into six different suits, **S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, and Elevation of Privilege (STRIDE). These different categories form the STRIDE model [15]. Each card is made up of a number, a suit and a description of a threat example corresponding to the suit of the card. The threat descriptions mostly for the benefit of new and/or inexperienced players, to provide helpful information and useful hints. Scoring is also a part of the game and it is meant to encourage competition and to promote flow and a sense of accomplishment, but the main aspect of the game is to bring some enjoyment to threat modeling, both for beginners and also experienced security experts. The Elevation of Privilege game starts by drawing a diagram describing the system that is to be modeled and dealing the cards between the players. The cards are played after the same suit and the threat linked to the system and for each threat, a point is added to the player who submitted the threat. At the end of the game the scores are tallied up and the one with the highest amount of points wins the game. [14]

2.2 Principles of Threat Poker

Secure software development means that the software team identifies relevant threat scenarios, and then removes vulnerabilities so that the identified threat scenarios are eliminated / blocked. Threat Poker assumes that the Scrum team is able to identify relevant threats that can negatively affect or take advantage of the functionality to be implemented during a sprint. This can be done by considering adversarial goals from the attacker's point of view, in the form of 'Misuse Cases' and 'Evil User Stories'. STRIDE [15] is a methodology that can assist team members in discovering threats. It is impossible to identify all relevant threat scenarios, and the Scrum team simply has to do it as best they can. Experience and expertise in threat modelling are important to be able to identify as many threats as possible.

Threat Poker consists of a *risk round* and a *solution round* for each relevant threat scenario. For each threat scenario, Threat Poker is played to:

1. Estimate both the security risk and the privacy risk resulting from the specific threat scenario.
2. Estimate the time and effort needed to remove the vulnerabilities exploited by the threat scenario, i.e. so that the threat scenario is blocked.

Each risk is due to a threat scenario, where a risk level is assessed by 1) how easy it is to execute the threat from the attacker's perspective, and 2) the negative consequence of the event, as shown in Figure 3. The risk level is calculated as a conjunctive combination of these two factors. This risk assessment exercise must be done separately for both security and privacy risks. In most

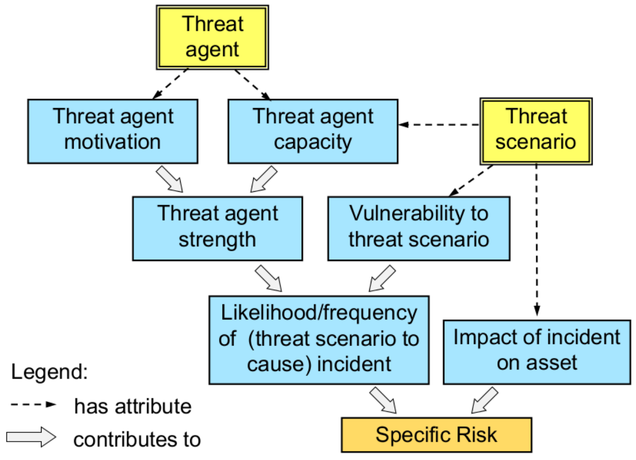

Fig. 3. Risk model for the factors of risk

cases, it would be natural to say that a particular threat scenario causes both security risk and privacy risk according to

$$\begin{cases} \text{Security risk} = (\text{ease of executing threat}) \times (\text{potential security impact}) \\ \text{Privacy risk} \;\; = (\text{ease of executing threat}) \times (\text{potential privacy impact}) \end{cases} \tag{2}$$

The basic principle estimating (security and privacy) risks is that the attacker's chances of success increases with the ease of executing a threat scenario, and that the resulting risk increases with the potential negative impact resulting from the threat scenario. If a new feature makes the system easier to attack, then the likelihood of an attack occurring will increase.

Figure 4 illustrates cards used for playing Threat Poker. Each player (member of the Scrum team) gets an entire suite from the deck, i.e. of Hearts, Spades, Diamonds or Clubs so that one card deck is sufficient for four players. For more than four players, two or more card decks are needed. The suit colour has no meaning other than separating the players from each other.

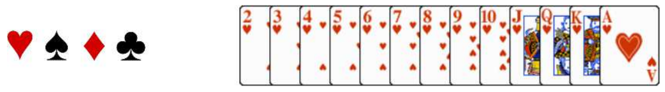

Fig. 4. Card Suits

First comes the *risk round* where each player can play two cards (face down), i.e. for security risk and privacy risk respectively. Low cards express low risk, and high cards express high risk. Then the cards are turned to show their values. In case of significant deviations between card values, a discussion follows where each player explains the reasoning behind the risk assessment. During the discussion the players typically influence each other's estimations. The risk round can then be repeated to converge risk estimates, and this pattern continues until an approximate consensus is achieved.

Then follows the *solution round* where the players can play two cards (face down) to estimate the effort of solving the threat, i.e. to remove vulnerabilities so that the threat scenario is blocked. Low cards express low effort and short time to implement the solution, while high cards express high effort and long implementation time. Once the players have played their cards, the cards are turned to show their values. A discussion then follows with repeated solution rounds until an approximate convergence emerges.

In the risk round, the level of security risk is represented by even values: 2, 4, 6, 8, 10 and Q (Queen). The level of privacy risk is represented by odd values: 3, 5, 7, 9, J (Jack) and K (King). Table 1 gives the interpretation of each card value in terms of security risk and privacy risk. The round starts by discussing the threat scenario, which can also be discussed during repeated risk rounds.

Table 1. Risk Levels

Even values: Security Risk Levels		**Odd Values: Privacy Risk Levels**	
2	Insignificant security risk, ignore	3	Insignificant privacy risk
4	Very low security risk, only solve if relatively low effort	5	Very low privacy risk, little or limited effect on data subjects
6	Low security risk, should be solved when moderate effort	7	Low privacy risk, transient or moderate negative impact on data subject
8	Moderate security risk, must be solved even when significant effort needed	9	Moderate privacy risk, long or significant negative impact on data subjects
10	High security risk, must be solved	J	High privacy risk, high and permanent negative effect on data subjects
Q	Very high security risk, with potentially very detrimental consequences	K	Very high privacy risk, with very serious permanent negative effects and potentially suicidal data subjects
A (Ace) – Extreme security risk or privacy risk. The project owner should seriously reconsider the viability of the user story of the present sprint.			

The benefit of letting Scrum team members first play their cards face down is that every member of the team initially is not influenced by any other team member, and is subsequently prompted to discuss his or her opinion. This principle will encourage less outspoken members to also share their expertise and knowledge with the rest of the team, but also to increase the general knowledge of the team. This will also help dealing with bias that may be introduced by outspoken and opinionated team members. Another benefit is to stimulate convergence towards teams consensus according to the principle of the Delphi Method [18].

Since a specific threat can cause both security risk and privacy risk, the team members can play both an even-value card and an odd-value card in the same round. The team needs to converge towards a consensus for both security risk and for privacy risk. Note that Table 1 lists the ace card ‘A’ as a special case, i.e. extreme risk either for security or privacy.

During the solution round, the estimated effort of implementing the solution as part of the present sprint is represented by the even values: 2, 4, 6, 8, 10 and Q (Queen). This type of solution is typically a non-functional security requirement, i.e. it is not a function that is part of the functional specification of the system. On the other hand, odd values: 3, 5, 7, 9, J (Jack) and K (King) express that the implementation of the solution is added to the backlog, i.e. it is not to be done as part of the current sprint. This type of solution is typically a functional requirement, i.e. it becomes a new item in the functional specification of the system. Table 2 describes the interpretation of each card value in terms of effort levels for the present sprint or to be part of a user story to be sent to the backlog. Whenever relevant, both cards with even numbers and odd numbers can be played. This round very similar to traditional planning poker. Before playing, the team starts with a discussion about possible solutions of the threat, which can also be discussed later during repeated solution rounds.

Table 2. Effort Levels

Even values: Sprint Effort Levels		**Odd Values: Backlog Effort Levels**	
2	Has already been solved	3	Will be solved as part of a user story in the backlog
4	Very low effort, very easy to solve in the present sprint,	5	Very low effort, very easy to solve, but should be done in a separate sprint
6	Low effort, relatively easy to solve in the present sprint	7	Low effort, relatively easy to solve, but should be done in another sprint
8	Moderate effort, relatively time consuming to solve in the present sprint	9	Moderate effort, relatively time consuming to solve in another sprint
10	High effort, the solution to this threat will take most of the time of most of the team in the present sprint.	J	High effort, the solution to this threat will be a major part of another sprint
Q	Very high effort, the solution will be so time consuming that the present sprint must be extended significantly	K	Very high effort, a separate sprint is needed to only focus on solving this threat
A (Ace) – Impossible to solve threat as part of the development project. The solution must be sought outside of the project.			

Since the solution of a specific threat can require components both as part of the present sprint and in the backlog, the team members can play both an even-value card and an odd-value card in the same round. The team needs to converge towards a consensus for both security risk and for privacy risk. Note that the ace card 'A' is a special case indicating that the solution is not a part of the project. This is a rule concerning security issues where the solution is out of the scope of the project and should be handed over to the appropriate people to be deal with.

After both the risk and the solution rounds have been played, the score of each threat will be calculated according to Eq.(3).

$$\text{Threat Score} = 2 \times \text{Risk Level} - \text{Effort Level}. \tag{3}$$

The decision to solve the threat should be set approximately when the threat score is greater than or equal to 14, i.e.

$$\begin{array}{ll} \textbf{IF} & \textbf{Threat Score} \geq 14 \\ \textbf{THEN} & \textbf{Solve threat} \\ \textbf{ELSE} & \textbf{Ignore threat} \end{array} \tag{4}$$

3 A Simple Threat-Poker Experiment

An experiment was conducted at the University of Oslo to observe the usefulness of Threat Poker, the best way to perform it, and the result that could be gained by implementing the method. Participants in the experiment were students in the Bachelor course IN2000 Software Engineering during the Spring semester 2018. The experiment was as part of the course where the students get extensive training in working in Scrum Teams.

During the planning phase of the experiment, an application was sent to NSD (Norsk Senter for Forskningsdata), for approval due to the participants being video recorded, which constitutes collection of personal data and thus is a privacy issue. The experiment was approved and the students gave consent to being video recorded, so the experiment could be conducted as planned.

The experiment was performed by several different student teams in the software engineering course. The teams participating in the experiment were comprised of between 2-6 students.

During the experiment it became evident that the students shared a lot of the same general knowledge, but with some of them having more specified knowledge about certain aspects of software security. While none of them were security experts, they all had thoughts and opinions about what should be required and how this could best be implemented.

During the session in which Threat Poker was played and practiced, all the participants took part in the discussion, asking questions, sharing knowledge, trying to think what could go wrong with the user story in case of an attack, how this could be exploited and how best to protect against the threat scenario.

The students were initially introduced to Threat Poker during an earlier lecture in the course. Immediately before each group participated in the experiment they were reintroduced to the method using a presentation about Threat Poker.

They were provided with the rules for the game, how it is played, the different rounds, the purpose behind it, and the general process. They were also provided with the interpretation of the different card values, and that even and odd values indicated whether they were dealing with a security risk or a privacy risk. This list and interpretation of card values – according to the participants – proved extremely helpful when it came to the estimations because by doing it this way, they got a guide on how their opinions should be estimated.

The students were given the same scenario that was to be discussed in the form of specific user stories that had been predefined to give each team the same starting point and to see the different groups under the same conditions and observe how the different groups worked with Threat Poker and the different observations they would make. The user stories were based on a specific hypothetical system that was to be developed, and the students were tasked with identifying and describing relevant threats to the system and how to deal with those threats.

The students were observed and filmed during the entirety of the Scrum meeting where they played Threat Poker. After the meeting they they were asked about their impression of the process, what went well and what could be improved to make the method more useful and intuitive. The observation of the students during the Scrum meeting provided qualitative indications of the advantages and disadvantages of playing Threat Poker as part of the meeting. During the Threat Poker sessions, the students could be interrupted to help clarify the different aspects of the process or information could be injected if and when needed.

At the beginning of the Threat Poker session, each team was given the same system description, the main requirements for the system, and the same user stories to work from. The hypothetical system to be developed was an online pharmacy e-commerce website. The user stories were set up in the form of: As a user, I should be able to .....

The team was given a list of several of them and tasked with coming up with the use case or misuse-case that could be associated with these user stories. The user stories involved several different scenarios, comprising mostly of:

- As a user, I should be able to log into the system.
- As a user, I should be able to create/edit my profile.
- As a user, I should be able to order product.
- As a user, I should be able to see my orders.

These user stories were set up in this way to provide a variety of security and privacy threats that are often present in every-day systems and thus needs to be addressed. For each user story, the teams were encouraged to come up with one main security and/or privacy threat and focus on that for the discussion.

After identifying and briefly discussing potential threats to the system or the users, the team started playing the risk round by playing cards face down to indicate their personal opinions about the severity of the security/privacy risks and then showed their cards. When all cards had been shown, and new discussion took place to explain the thought process behind the individual estimations of each player. This gave all the players a chance to discuss the threat, and to share their opinions about how severe this threat could be if allowed to remain unchecked. This brought to the surface new information that had previously not been discussed, and allowed the students with limited knowledge to learn from more students with more knowledge and experience.

Since Threat Poker is a method which encourages those players with the highest and lowest risk/effort estimates to explain their reasoning to each other it stimulates all members of the team to share their opinions. This effect could clearly be observed during the experiment.

In the discussion during the initial round, some students were relatively reluctant to share their thoughts, but as the game progressed, and the card values were shown, each member got to express his or her opinion and take part in the discussion. The risk round was repeated a few times until the group reached a consensus on the security and privacy risk levels. Then the game progressed to the solution round which started with a discussion about how best to solve this threat, and then by playing with cards to express the estimated effort required to implement the solution. Figure 5 shows a student Scrum Team playing Threat Poker during the experiment.

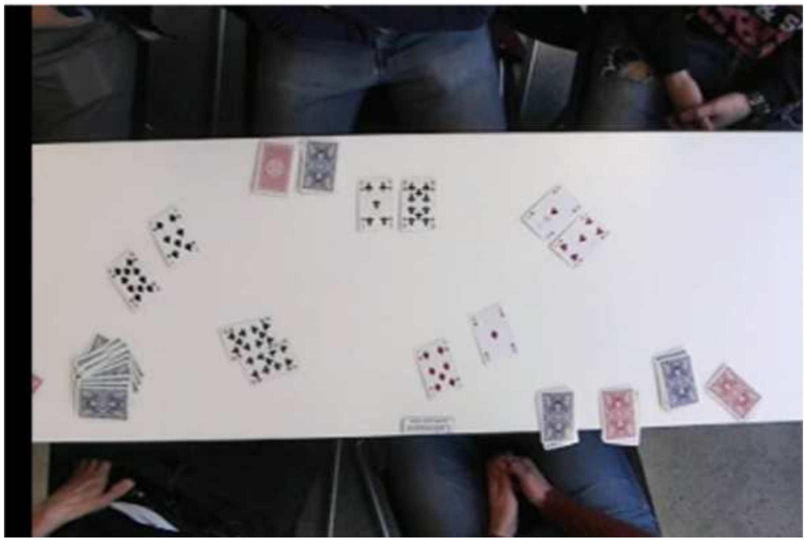

Fig. 5. Actual round with participants and cards playing Threat Poker

For the students, a consensus was generally reached rather quickly, typically requiring no more than two repetitions of the risk round before proceeding to the solution round. From observing the game it seemed that the largest amount of time went into discussing and understanding each specific threat scenario that was going to be subject of the risk and solution rounds in the Threat Poker game. To test this assumption, the last group of students was presented with specific threat scenarios for the system to be developed, so they did not need to identify and discuss threat scenarios. This proved to be more efficient in some ways, in that it allowed for more threat scenarios to be considered as part of the Threat Poker game. When all relevant threat scenarios for one user story had been considered, the group could jump to the nest user story and play Threat Poker over new threat scenarios. The last group was therefore able to plat Threat Poker with more user stories than the other groups had.

Doing it this way still allowed for many specific threats to be discussed, but not for new threats to be discovered by the team.

4 Discussion and Conclusion

Using Threat Poker helps estimating the severity of security risks and/or privacy risks identified during software development, as well as estimating the effort to solve those risks. This can be seen as an important element of privacy-by-design and security-by-design which is now required for the development of computer software.

The general consensus from all the student groups was that Threat Poker is a useful technique, and helpful to generate a discussion about security that should be a mandatory part of the development process. Using Threat Poker forced the groups to consider different threat scenarios and how to deal with them.

By giving the groups the same user stories, and by them having much of the same knowledge and experience, resulted in many of the same threats, and also much similarity in the suggested solutions by the different groups. The estimations provided by the team members were somewhat different inside each group, but after a few repetitions of a round, they reached a consensus. When reaching a consensus, the estimation differed from group to group, but was generally in the same range for the similar threats.

While generally a very helpful technique, several different ways for improving Threat Poker was suggested by the different groups. Chief among them was a suggestion for the solution part of the card game.

The first groups were given the instruction that in the solution round, the card value would indicate the ease of implementing the solution to the threat, meaning that value 2 would indicate that the threat would be difficult to solve, and that value 12 (Queen) would indicate that the threat would be easy to solve. The students in the first group found this confusing, and indicated that to make Threat Poker more intuitive, the solution estimation with the cards should be the opposite of how they were told to play, meaning that a low card value should indicate that the solution required low effort to implement, and a high card value should indicate high effort to implement the solution.

The convention suggested by the students was tried with the last group, and from their perspective, this seemed like the easier and more intuitive way of expressing effort during the solution round of Threat Poker, and is exactly how we have described threat poker above in this paper.

References

1. Zulkarnain Azham, Imran Ghani, and Norafida Ithnin. Security Backlog in Scrum Security Practices. Technical report, Universiti Teknologi Malaysia, 2011.
2. Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Principles behind the Agile Manifesto. http://agilemanifesto.org/iso/en/principles.html, 2001.
3. Microsoft Corporation. SDL: Microsoft Security Development Lifecycle. Version 4.1., 2009.
4. Microsoft Corporation. Security Development Lifecycle for Agile Development. Ver. 1.0., 30 June 2009.
https://www.microsoft.com/en-us/SDL/Discover/sdlagile.aspx.
5. James Grenning. Planning Poker or How to avoid analysis paralysis while release planning. Technical report, Wingman Software, 2002.
6. Shon Harris and Fernando Maymí. *CISSP All-in-One Exam Guide, 7th Edition*. McGraw-Hill, 2016.
7. Richard Kissel et al. Security Considerations in the System Development Life Cycle – NIST Special Publication 800-64, Rev. 2. Technical report, National Institute of Standards and Technology, October 2008.
8. Nabil Mohammed, Ali Munassar, and A. Govardhan. A Comparison between Five Models of Software Engineering. *Int. Journal of Computer Science Issues (IJCSI)*, 7(5), September 2010.
9. OWASP. ASVS - Application Security Verification Standard v.3.0.1 2016, 2016.
10. Christoph Pohl and Hans-Joachim Hof. Secure Scrum: Development of Secure Software with Scrum. Technical report, Munich University of Applied Sciences, 2015.
11. QASymphony. Agile Methodology: The Complete Guide to Understanding Agile Testing. https://www.qasymphony.com/blog/agile-methodology-guide-agile-testing/, 2017.
12. Ken Schwaber and Jeff Sutherland. The Scrum Guide, 2017.
13. G. Shipley, Meneely A., and L. Williams. Protection Poker: The New Software Security “Game”. In *IEEE Security and Privacy*, pages 14–20, 2010.
14. Adam Shostack. Elevation of Privilege: Drawing Developers into Threat Modeling. https://www.microsoft.com/en-us/download/details.aspx?id=20303, 2012.
15. Adam Shostack. *Threat Modeling: Designing for Security*. Wiley Publishing, 1st edition, 2014.
16. VersionOne. Agile 101 General Learnings. https://www.versionone.com/agile-101/.
17. Dave Wichers. Breaking the Waterfall Mindset of the Security Industry. In *OWASP AppSec USA*, New York, 2008.
18. Laurie Williams and Andrew Meneely. Protection Poker: The New Software Security ”Game”. Technical report, North Carolina State University, 2009.