Threat Poker: Gamification of Secure Agile *

Audun Jøsang, Viktoria Stray, and Hanne Rygge

University of Oslo, Norway, {josang,stray}@ifi.uio.no
Norsk Tid AS, hanne.rygge@gmail.com

Abstract. Agile software development is practiced in most software development projects around the world. To explicitly consider and include security requirements as part of agile software development is referred to as 'secure agile'. To include security will naturally require additional time and effort, with potentially reduced agility as a consequence. To maintain agility, it is important to have efficient methods to include security in the development process. In this study, we describe enhancements to Threat Poker, which is a game designed for the software development team to deal with security threats identified during the agile development project. Games can be valuable educational tools for actively engaging students and practitioners alike. An experiment with students indicates that playing Threat Poker increases security awareness and that it is a fun and simple way to discuss identified security threats and how to remove security vulnerabilities during the software development process.

1 Introduction

Agile software development, currently the most common approach to software development, was not originally conceived with security in mind. As such, many agile projects do not have built-in steps for dealing with security issues. However, with the introduction of new laws and regulations for security, such as GDPR in the EU, design principles for 'security by design' and 'privacy by design' must be followed in order for the developed systems to be legally compliant [17].

'Security by design' and 'privacy by design' are relatively general concepts for describing the integration of security and privacy considerations throughout the life-cycle of ICT systems. The term 'by design' simply means that threat, vulnerability and risk assessment related to information security and privacy must be included in the requirements specification, architecture design, coding, testing, deployment and management of all systems, products and services that involve ICT components. This paper focuses on using gamification to strengthen security and privacy by design during the steps of architecture design and coding.

Gamification [1] is defined as *"The application of typical elements of game playing (e.g., point scoring, competition, rules of play) to other areas of activity, typically as an online marketing technique to encourage engagement with a product or service"*. The principle of gamification is to use various aspects of games

* Published in the proceedings of the 13th World Conference on Information Security Education (WISE13). Maribor, Slovenia, September 2020.

to engage the 'players' in the task at hand. Students of software engineering are generally positive to using games to stimulate learning [8]. A recent systematic literature review concluded that gamification in software engineering education is still in its infancy [2]. The authors encourage researchers to explore this topic further since gamification is shown to motivate students and stimulate learning.

Several researchers have introduced games to increase awareness of security in software development projects. For example, Denning *et al.* proposed a tabletop card game called Control-Alt-Hack to generate awareness of security issues and increase the accuracy of people's perceptions of computer security as a discipline [5]. They found that students enjoy playing the game and that it increases their security awareness. Another game called [d0x3d!] is an open-source content-license game (allowing free distribution and adaptation) that provides an artificial context for discussing real ideas in network security [6]. The Security Requirements Educational Game (SREG) is a card-based game that is intended to be used for security-requirements education and that has shown to increase the understanding of security issues and vulnerabilities [24].

Williams *et al.* introduced Protection Poker as a game for estimating security risk during software development projects [22, 23]. Protection Poker is inspired by Planning Poker (see Sec.3.3 below) and uses special cards to express levels of risk resulting from a specific threat. The effect and usability of Protection Poker were further studied in [18, 19] where it was found that challenges for its adoption are the overhead in terms of time spent on playing Protection Poker, as well as the difficulty of transforming results from playing Protection Poker into more secure code. This calls for the need to make the game light-weight and to focus on solutions that have a positive impact on making the code more secure.

Threat Poker introduced in 2018 by Rygge and Jøsang [12] is inspired by Protection Poker in the sense that it also focuses on estimating risks related to threats, similarly to Protection Poker. In addition, Threat Poker also focuses on estimating the effort of implementing security solutions during agile software development. An important goal of Threat Poker is to make the game easy to learn and play for students, and for that reason it only uses standard playing cards. Using a deck of familiar playing cards (instead of unfamiliar specialized cards) is a good way of giving an intuitive feeling of playing a game during learning, and during practice in a software development team.

This study proposes ways to improve Threat Poker and investigates how it fits into an agile development process. For this purpose we investigated the following research questions:

RQ1: How can threat modelling and risk assessment be included in agile software development?

RQ2: How can Threat Poker be adapted and utilized in agile projects?

The remainder of the paper is organized as follows: Risk management and threat modelling are briefly described in Section 2, then the principles of agile software development are described in Section 3. In Section 4, we describe our research method and results of adapting Threat Poker. In Section 5, we discuss our findings, conclusion and suggest future work.

2 Risk Management

2.1 Threat Modelling

Threat modelling is to identify how systems can be attacked, which is an essential step in identifying vulnerabilities and risks. *"Threat modelling works to identify, communicate, and understand threats and mitigations within the context of protecting something of value."* [15].

The main purpose of doing threat modelling is to identify relevant threat scenarios with the aim of understanding how they can be mitigated or blocked.

Threat modelling is a component of risk management. The steps of risk management are typically to define the target scope of the risk management, identify the threat agents and possible threat/attack scenarios, understand existing countermeasures and their limitations, identify remaining exploitable vulnerabilities, estimate and prioritize identified risks, and last to propose and implement countermeasures to mitigate and remove vulnerabilities which in turn results in strengthened security, and hence in reduced risk.

2.2 Threat Actors and Threat Scenarios

It is important to distinguish between the concepts of 'threat actor' and 'threat scenario', where the former denotes active agents (e.g., persons or organisations), and the latter denotes what they do as a series of steps during an attack. When simply using the term 'threat' in this paper, it should be understood as a 'threat scenario'.

Many threat actors have the motivation and capacity to attack digital systems and ICT infrastructures. It is crucial to understand and attempt to predict how threat actors will attack, which is done by threat modelling. Identifying threat scenarios is a prerequisite for knowing how to stop them. It is, of course, impossible to identify all relevant threat scenarios. However, the designers and developers of software products must use the security skills and experience they have to do the best they can. Additional training or the inclusion of more experienced persons can be required if the software development team members feel that they have insufficient security skills and experience.

The identification of threat scenarios requires the team members to think like an attacker, and try to come up with a method to attack the system. Whenever the team sees an opportunity to successfully attack the system they have identified a threat scenario. The difference between 'threat scenario' and 'attack' is that an attack can be seen as the instantiation of an abstract threat scenario. There can be an infinity of different threat scenarios, but only those that are practical and realistic should be considered. When a threat scenario has been identified which realistically could be executed, it means that there are vulnerabilities in the system or in the surrounding infrastructure that can be exploited by potential attackers to execute the threat scenario. Blocking or mitigating a threat scenario is equivalent to removing the corresponding security vulnerabilities, and this is precisely what has to be done by the design team during software development.

3 Agile Software Development

Agile software development is an umbrella term which covers different frameworks and practices which can be applied by software developers to deal with a continuous changing environment, based on specific principles that were outlined in the Agile Manifesto [4]. The main principles consist of early and continuous delivery, welcoming changes, frequent delivery of working software, business people and developers working together, building projects around motivated people, and giving them the resources to succeed [3].

Agile focuses on face-to-face meetings, seeing working software as the measurement of progress and also on promoting sustainable development. Agile also gives attention to technical excellence and good design, simplicity, and assumes that self-organizing teams help develop good architectures, requirements and design. It also has a focus on reflection and on adjusting the process accordingly.

3.1 Scrum

Scrum is the most widely used agile development methodology [21] and is set up by fixed rules and roles for the team members [13].

The Scrum process consists of several different components such as the product backlog, sprint planning, daily Scrum meetings, review meetings and retrospective meetings. Like most development methodologies, Scrum originally was not designed with a specific integrated way to deal with security and privacy concerns, and because of this it does not contain a guide on how to best implement security in the method. There have been developed several methods to try and solve this problem like 'Secure Scrum', and 'Security Backlog' in Scrum. Authors have also developed games that can be useful for dealing with the security shortcomings of Scrum. Examples of games are Protection Poker [23], Microsoft's Elevation of Privilege [14], as well as Threat Poker described here.

3.2 Secure Scrum

The Secure Scrum Model [20] is an example of how to consider security in Scrum. Its four main components are: 1) the Identification component, 2) the Implementation component, 3) the Verification component, and 4) the Definition-of-Done component. These components have the effect of influencing the different stages in the Scrum process. Security concerns are identified and marked in the Product Backlog as a result of the Identification component. The Implementation component is used in Sprint Planning and the Daily Scrum meetings and is focused on the awareness of the security concerns. The possibility of testing with focus on security is done in the Verification component. Finally, the Definition-of-Done component defines what Definition-of-Done for security related issues are.

When dealing with the security backlog of Scrum in Secure Scrum, a new backlog can be added to deal with all the new security concerns the occurs in the software or due to new features added to the software solution [20]. The purpose of the security backlog is to implement different design principles based

on security and privacy to limit the numbers of vulnerabilities and to reduce the risk to the software that is being developed. In addition to the new backlog, another role can be added, called the Security Master, whose role is to manage the security backlog and also decide which security or privacy features require the most attention, and then add it to the sprint backlog which is then completed by the developers.

3.3 Planning Poker

Planning Poker [7] is a card-based method for time estimation and planning, often used when following the Scrum process. Planning is started by reading a user story or description of the feature to be developed. Each of the team members will have a deck of special cards, Planning Poker cards, which are cards with increasing values, often the sequence: 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100. The team members are then able to ask questions of the product owner and discuss the feature internally. When the team finishes the discussion, each member will use the cards for estimation, and if the team members are in agreement, the estimate becomes the estimate for the feature. If there are discrepancies in the estimation, a second round will commence, with new discussions and estimations, and this pattern will continue until a relative consensus is reached.

3.4 Kanban

Kanban was developed for Toyota's Lean Production System as a specific approach to agile software development and is inspired by the requirements for just-in-time production systems. Kanban focuses on visualizing the 'what', the 'when' and the 'how' of the production process, or development process when talking about software development. There are four major principles that make up the Kanban framework. These consists of visualizing work to increase communication and collaboration, avoiding an unmanageable amount of non-prioritized open tasks, measure and optimize the flow, gather information, and predict future problems, and lastly, to aim for continuous improvement by analysis [9]. Teams that employ Kanban often use practices such as daily meetings and retrospective meetings, but rarely Planning Poker as they do not have to estimate the backlog items the same way as is needed in Scrum (when planning the next iteration).

4 Threat Poker

Threat Poker is an efficient method for discussing threats in agile software projects and how to remove or mitigate vulnerabilities in the developed software [12]. Two essential elements of security by design are that the designers identify relevant threat scenarios, and then take the necessary steps to block or mitigate the said threat scenarios, i.e., to solve the threats. In this paper we focus on Threat Poker as a form of 'gamification' of the process for dealing with threats and vulnerabilities during agile software development.

4.1 Equipment

In Threat Poker, the team members use regular decks of playing cards. Most people are familiar with traditional playing cards, and sitting around a table with playing cards in the hand typically triggers the positive feeling of playing a game. People's familiarity with playing cards also gives an intuitive interpretation of the cards' face values with regard to threats and their solutions. Low cards intuitively represent low threats or effort estimations, and picture cards intuitively represent high threats and effort estimations for solving the threat scenarios.

4.2 Adaptation to Agile Methods

Depending on the development method, there are several stages in the used methodology where Threat Poker can be implemented. When using Scrum, several specified meetings are used for different purposes. There are daily Scrum meetings, planning meetings, and also retrospective meetings. When using Scrum, Threat Poker is most useful in the planning stages of the process where the sprint iteration is planned out, what feature to work on, where security and privacy risks can be assessed and evaluated and where the solution can be discussed, found and solved, if needed, during each sprint.

Kanban, on the other hand, does not have specified meetings set up in the methodology, but a common practice is to use retrospective meetings to improve the team's way of working, and participants often discuss past challenges and how to overcome them to work better together in the future. We suggest that a retrospective in agile software development could be a good meeting to play Threat Poker in since it is meant to be a positive team meeting which would fit with playing a game.

We suggest that Threat Poker can be used as a means to identify and discuss threats to be worked on, and that the next step would be to make these threats visible to the whole project, for example by introducing a separate 'Security board' or 'Threat board' or as tasks added to the backlog or the 'todo'-column. The idea is that when, during the development process, a security or privacy concern is discovered, it could be added directly to the security board and then, during either the planning meeting or the retrospective meeting, could be discussed and made into a features and added to the 'To-Do' column of the Kanban board.

Agile teams also rely on daily meetings to coordinate their work and make decisions [16], we therefore suggest that identification of threats could be part of the daily meetings. If a person has identified a threat, this can be discussed in the daily team meeting. It could be added to the board or backlog. Either as a security item to be investigated or the team could have a designated Security or threat board were they could collect possible threats and discuss these when playing Threat Poker. Discussing security threats would then be an iterative process, which fits well with agile methods.

4.3 User Stories and Use Cases vs. Attacker Stories and Threat Scenarios

A user story describes the user, what the user is supposed to do as well as what the user wants to achieve. A user story typically is in the format: *"As a* **user***, I want* **action** *so that* **effect***"*. This approach can also be applied to the attacker which can be expressed as an 'attacker story' e.g. expressed as *"As an* **attacker***, I want* **to execute the threat scenario** *so that* **I reach my attack goals***"*.

A use case is a more detailed description of a function or application than a user story. A use case includes actions or event steps typically defining the interactions between roles (actors) and system components to accomplish a goal. The negative version of a use case is a 'threat scenario' which includes actions or event steps typically executed by the threat actor (attacker) who misuses system and network components and interactions between them to accomplish the attack goal. A threat scenario can thus be seen as a 'misuse case'.

The features to be discussed during Threat Poker can be presented either as an attacker story, as a threat scenario, or as a specific vulnerability without any specific attacker story or threat scenario associated with it.

4.4 Vulnerability Management

The Open Web Application Security Project (OWASP) describes the 'Top 10 Security Risks' [11]. These risks involve well-known security threats and corresponding vulnerabilities that (unfortunately) are typically found in web application software. Similarly, the Common Weakness Enumeration (CWE[1]) is a community-developed list of common software security weaknesses (vulnerabilities). CWE serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts. CWE and OWASP Top 10 are a good starting point for discussions around threats and vulnerabilities during software development. Security vulnerabilities in software can also be identified and assessed without reference to specific threat scenarios. For example, CVSS (Common Vulnerability Scoring System) provides a method for estimating the severity of specific security vulnerabilities. The severity level can be translated into a qualitative representation (such as low, medium, high, and critical) for prioritization in vulnerability management.

4.5 Estimation of Risk and Effort

Threat Poker focuses on evaluating the risks from relevant threat scenarios or other vulnerabilities that the team is able to identify, and on estimating the efforts needed to solve those threats and vulnerabilities. The face values of the playing cards are thus used to represent these two things:

- the estimated risk level of a specific threat scenario or vulnerability,
- the estimated effort level to solve the specific threat scenario or vulnerability.

[1] https://cwe.mitre.org/

Risk-Level Estimation. There is a distinction between security risk as seen from the point of view of the organisation, and the privacy risk as seen from the user (data subject in the terminology of GDPR). Threat Poker considers both types of risk together.

Two main factors or principles are considered for estimating security/privacy risk levels. The first principle states that risk increases as a function of the probability of attack, i.e. the ease with which the attacker can successfully execute the threat scenario. The second principle states that risk increases as a function of the negative security/privacy impact resulting from an attacker's successful execution of the threat scenario.

Security risk is the negative impact affecting the owner of the information assets that are being attacked. Privacy risk is the negative impact affecting a person whose personal information has been misused with reference to relevant data protection principles such as those of GDPR. This can be expressed as:

$$\begin{cases} \text{Security risk} = (\text{ease of executing threat}) \times (\text{potential security impact}) \\ \\ \text{Privacy risk } = (\text{ease of executing threat}) \times (\text{potential privacy impact}) \end{cases} \tag{1}$$

Given the different victims of security risk and privacy risk respectively, there can be different threat scenarios causing these different risks. To consider these two types of risks separately might help the team analyze and prioritize the most important and pressing features to implement and how to best deal with the security and privacy concerns regarding the features. Note that the system owner can represent a privacy threat actor in case it collects and processes personal information in breach of data protection principles.

Identifying and estimating these risks can be difficult, especially when the team consists of developers who may not have much experience with software development that focuses on security. There are several standards that can aid the team in finding risks or give an indication of where the most common security and privacy risks occur and also how to deal with them.

In software and system development, the principle of 'security by design' means to adequately consider security during every step of the development process, and when working with this, it can be helpful to consider common design principles. Several principles are defined by OWASP (Open Web Application Security Project) in its ASVS (Application Security Verification Standard) [10], and these can give an indication of most common vulnerabilities, and also guidelines on how to prevent these risks.

Effort-Level Estimation. The estimation of the effort needed to solve the threat or vulnerability is similar to the estimation of effort levels in Planning Poker described in Section 3.3 above. The solution to the identified threat can be seen as a new item to be added to the backlog in Scrum, or to the board in Kanban. This item can be taken into the current development sprint, or be delayed to a subsequent sprint. In any case, the effort to implement the idem should be estimated by the members playing Threat Poker.

4.6 Card Values

Figure 1 illustrates suits of cards used for playing Threat Poker. Each player (member of the Scrum or Kanban team) gets an entire suite from the deck, i.e. of Hearts, Spades, Diamonds or Clubs. With four suits one 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.

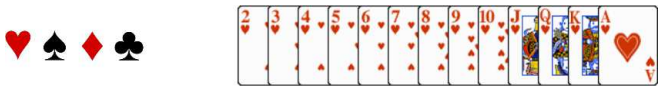

Fig. 1. Card Suits

Each player sorts out the cards, odd-numbered cards and even-numbered cards. The security and/or privacy risk is represented by odd values: 3, 5, 7, 9, Jack (11), King (13) and A (Ace). The solution-effort level is represented by even values: 2, 4, 6, 8, 10 and Queen (12). As a mnemonic, players of Threat Poker can see risk as an 'oddity' (represented by an odd-value card) which must be 'evened out', i.e. fixed (represented by an even-value card).

Table 1 gives the interpretation of each card value in terms of security/privacy risk and effort level to solve the threat.

Table 1. Risk and Effort Levels

Odd Values: Risk Levels	**Even Values: Effort Levels**
3 Insignificant risk, can be ignored unless the effort is minimal.	2 Minimal effort, can be solved quickly and easily.
5 Very low security and/or privacy risk, only solve if the effort is (very) low.	4 Very low effort, very easy and quick to solve.
7 Low security and/or privacy risk, should be solved if the effort is low/moderate.	6 Low effort, relatively easy to solve
9 Moderately high risk, should be solved even if (moderately) high effort needed	8 Moderately high effort, solve threat in this sprint if time, or else in other sprint
J High risk, should be solved even when high or very high effort needed	10 High effort, the solution to this threat will be a major part of this/other sprint
K Very high risk, with potentially very severe consequences. Must be solved.	Q Very high effort, a separate sprint might be needed to focus on solving this threat
A (Ace) – Extreme risk. Must be solved, or else reconsider the viability of user story.	

Threat Poker described in [12] used a different interpretation of card values, and had two different types of rounds for playing a game. First there was a risk estimation round which was followed by a solution-effort estimation round. We found that the relative complexity of this scheme in itself was a barrier to learning and for successfully practicing secure agile.

We simplified the original Threat Poker from [12] so that participants are playing integrated risk and solution rounds instead of separate rounds for the risk level and the effort level respectively. In the integrated version, odd-numbered cards now indicate the (security and privacy) risk level associated with the threat, while the even-numbered cards indicate the estimated effort required to solve the threat, i.e. to remove or mitigate the corresponding vulnerability.

4.7 Playing Threat Poker

For each user story to be implemented the team must do threat modelling as described in Section 2.1. Threat modelling is a prerequisite for secure agile, and for applying Threat Poker.

A game of Threat Poker starts by discussing a specific threat scenario or vulnerability which has been identified. This discussion can continue during the repeated rounds of the same game.

For each threat or vulnerability that the team has identified and wants to consider, they play a game of Threat Poker as illustrated in Figure 2.

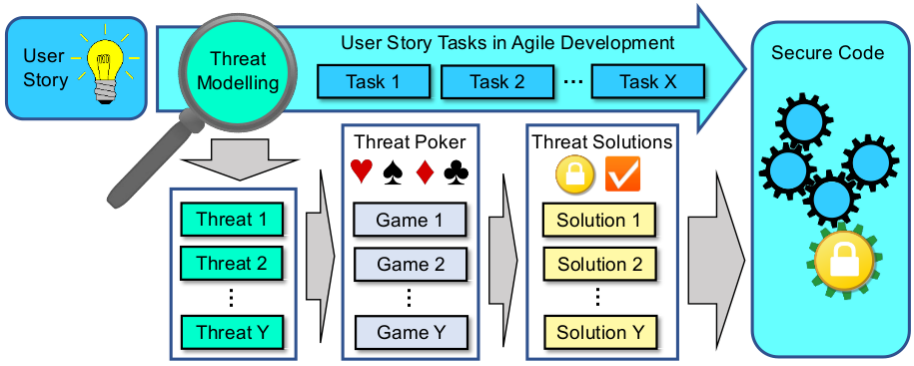

Fig. 2. Integration of Threat Poker in secure agile

The game starts by discussing the security and privacy risks involved with the threat, as well as the best solution to solve the threat, i.e. to remove the vulnerability. When the preliminary discussion is completed, each player must make two subjective estimations; one for security/privacy risk and one for the effort/time needed to solve it. Then they play out cards for the first round.

Each player puts down two cards facing down; an odd-value card for the risk level and an even-value card for the solution effort level. Low cards express low risk/effort, and high cards express high risk/effort. Then the cards are turned to show their values. In the case of significant deviations between card values, a discussion follows where each player explains the reasoning behind the risk and/or effort assessment. During the discussions, the players typically influence each other's estimations. Then a new round is played.

Repeated rounds are played, with discussions in between, until an approximate consensus is achieved, for both risk and effort levels. After a consensus is reached, the estimated risk and solution effort levels are used as parameters in the simple decision rule of Eq.(2) as an approximate guideline to decide whether the team should spend time and effort to solve the threat.

$$\begin{array}{ll} \textbf{IF} & \textbf{Risk level} - \textbf{Effort level} > 0 \\ \textbf{THEN} & \textbf{Solve threat} \\ \textbf{ELSE} & \textbf{Ignore threat} \end{array} \tag{2}$$

Threat Poker is thus also an instrument to assist the team in making the decision whether a threat should be solved. The team must decide if the solution shall be implemented in the current iteration or be put into the backlog, or even if it is better to reconsider the viability of the whole user story.

We had students at the University of Oslo trying out Threat Poker. The course was a 20 ECTS course involving a major project in software engineering. During a project period of 13 weeks, students were assigned to develop an app and using agile methods and practices. We had student teams play Threat Poker and report how they experienced it. The teams participating were comprised of between 2-6 students.

The simplification of Threat Poker by integrating separate rounds of risk and effort estimation into one round also meant that the participants just played a single odd-valued card to represent a combined risk level for both security and privacy, and a single even-valued card to represent the effort for solving the risk. At this point in the game, the specific type of risk (security or privacy) is considered irrelevant. The important point is that threats and risks should be solved anyway, regardless of whether it is a security risk or a privacy risk. When revealing the card values, each developer explained whether the risk value represented by the odd-valued card was a security risk, a privacy risk, or both.

4.8 Traditional Playing Cards vs. Planning-Poker Cards

There are several different games used for development purposes, and many of them card games. Common among them is that they often require or encourage the use of specially developed cards to play the game. Planning poker use planning poker cards with values based on a Fibonacci sequence, Microsoft's Elevation of Privilege also use specific cards to describe the risk to be played about, Threat Poker, on the other hand, when under development, it was decided to use regular playing cards.

There exists several different reasons why this was thought the best tool to use for estimation in this game, but reasons that could also be applied to other games intended for development teams.

By using playing cards, the team members are provided with cards they are already familiar with, and with face values that are intuitive to the players because they already know how the cards are used for playing. The use of regular playing cards with familiar face values give better intuitive meaning than just a

set of values in a number sequence. Based on the intrinsic value of the different types of cards, the risk severity expressed by the value of a king which is a high value card in a regular card deck, might be easier to understand than the value 50 in a number sequence, because familiar playing cards provide more of a context to the values than what unfamiliar cards would provide.

Using regular playing cards also helps creating an association between traditional card games and Threat Poker, and as such can help make the relatively abstract tasks of risk evaluation and effort estimation into something fun and tangible. Gamification around the discussion and estimation of threats and solutions will also stimulate learning among the members of an agile software team.

Another reason for using regular playing cards is their availability. While specially designed cards, such as the Planning Poker cards, and the cards used in Microsoft's Elevation of Privilege can be difficult and time consuming to get hold of, normal playing cards are easy and cheap to buy anywhere. Getting regular playing cards does not require special ordering or manufacturing, and most offices will have at least one deck lying around, ready to be used.

4.9 Variations of Threat Poker

Another way of playing Threat Poker is as a board game. Some modifications are required to the method on how to play, but not on the fundamental principles of the game. The team will still use the same user stories and discussion techniques, but will forgo the deck of cards in favor of a created 2x2 grid that can easily be drawn up before the game is stared. The grid is drawn up and each sector will represent a different level of risk and effort. One sector will be high risk and difficult to solve/long time to solve, another will be high risk but easy to solve/short time to solve, the third will be low risk and difficult to solve/long time to solve and the last is low risk and easy to solve/short time to solve. Each player will use some form of tokens, to place on the board, and after each discussion or round, will move their tokens to represent their opinions on the feature that is discussed. When all players are in agreement with tokens in the same section of the grid, the team has reached consensus and can move on.

5 Discussion and Conclusion

Threat Poker is a card-based method to help development teams focus on security and privacy by design which is becoming a standard requirement for software development, as e.g. required by law in the EU due to GDPR. Threat Poker helps guide developers into considering security and privacy threats and vulnerabilities during the entire development process and also in producing documentations for privacy and security audit.

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 are now required for the development of computer software.

We had student teams in a software engineering course at the University of Oslo play an adapted version of Threat Poker. Our results show that even though the students shared a lot of the same general knowledge, and none of them were security experts, Threat Poker helped articulate thoughts and opinions about possible threats to the system. 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 general feedback from all the student teams 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 teams to consider different threat scenarios and how to deal with them. We believe that the simplified version of Threat Poker described in this paper, a threat and its solution are discussed and evaluated in the same round, is better than the original version of Threat Poker where threats and solutions were discussed in separate rounds.

Future research should investigate how Threat Poker encourages and supports teams to solve security threats, compared to other games that exist such as the Security Requirements Education Game [24], [d0x3d!][6] and Control-Alt-Hack [5]. Furthermore, since we have only tested Threat Poker on students, future research should investigate how agile software development projects in companies would benefit from Threat Poker, and in what way the developed software becomes more secure as a result of playing the game.

References

1. Definition of gamification. https://en.oxforddictionaries.com/definition/gamification.
2. Manal M Alhammad and Ana M Moreno. Gamification in software engineering education: A systematic mapping. *Journal of Systems and Software*, 141:131–150, 2018.
3. 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.
4. Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001.
5. Tamara Denning, Adam Lerner, Adam Shostack, and Tadayoshi Kohno. Control-Alt-Hack: the design and evaluation of a card game for computer security awareness and education. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 915–928. ACM, 2013.
6. Mark Gondree, Zachary NJ Peterson, and Tamara Denning. Security through play. *IEEE Security & Privacy*, 11(3):64–67, 2013.
7. James Grenning. Planning Poker or How to avoid analysis paralysis while release planning. Technical report, Wingman Software, 2002.

8. G Ivanova, V Kozov, and P Zlatarov. Gamification in Software Engineering Education. In *42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1445–1450. IEEE, 2019.
9. H Kniberg and M Skarin. Kanban and Scrum making the most of both. http://www.infoq.com/minibooks/kanban-scrum-minibook, 2010.
10. OWASP. ASVS - Application Security Verification Standard v.4.0, 2019.
11. OWASP (Open Web Application Security Project). Threat modeling. https://owasp.org/www-community/Threat_Modeling, 2020.
12. Hanne Rygge and Audun Jøsang. Threat Poker: Solving Security and Privacy Threats in Agile Software Development. In *The 23rd Nordic Conference on Secure IT Systems (NordSec 2018)*, Oslo, 2018. Springer.
13. Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.
14. Adam Shostack. Elevation of privilege: Drawing developers into threat modeling. In *2014 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.
15. Adam Shostack. *Threat modeling: Designing for security.* John Wiley & Sons, 2014.
16. Viktoria Stray, Nils Brede Moe, and Dag IK Sjøberg. Daily Stand-Up Meetings: Start Breaking the Rules. *IEEE Software*, 2018.
17. Colin Tankard. What the GDPR means for businesses. *Network Security*, 2016(6):5–8, 2016.
18. Inger Anne Tøndel, Martin Gilje Jaatun, and Daniela Soares Cruzes. "collaborative security risk estimation in agile softwarer development". *Information and Computer Security*, 27, 2019.
19. Inger Anne Tøndel, Martin Gilje Jaatun, Daniela Soares Cruzes, and Tosin Daniel Oyetoyan. Understanding challenges to adoption of the protection poker software security game. In Katsikas. S. et al., editors, *Computer Security, ESORICS 2018 International Workshops: SECPRE 2018, CyberICPS 2018, LNCS 11387*. Springer, Cham, 2018.
20. Sven Törpe and Andreas Poller. Managing security work in Scrum: Tensions and challenges. In Martin Gilje Jaatun, editor, *International Workshop on Secure Software Engineering in DevOps and Agile Development (SecSE 2017)*, Oslo, 2017.
21. VersionOne. 12th state of agile report. https://www.infoq.com/news/2018/04/state-of-agile-published.
22. L. Williams, M. Gegick, and A. Meneely. Protection Poker: Structuring Software Security Risk Assessment and Knowledge Transfer. In *International Symposium on Engineering Secure Software and Systems*, pages 122–134. Springer, 2009.
23. Laurie Williams, Andrew Meneely, and Grant Shipley. Protection poker: The new software security" game". *IEEE Security & Privacy*, 8(3):14–20, 2010.
24. Affan Yasin, Lin Liu, Tong Li, Jianmin Wang, and Didar Zowghi. Design and preliminary evaluation of a Cyber Security Requirements Education Game (SREG). *Information and Software Technology*, 95:179–200, 2018.