

Agile Security Requirements Engineering

Johan Peeters

Independent, yo@johanpeeters.com

Abstract

Agile processes have been deemed unsuitable for security sensitive software development as the rigors of assurance are seen to conflict with the lightweight and informal nature of agile processes. However, such apparently conflicting demands may be reconciled by introducing the new notion of abuser stories in the requirements domain. These extend the well-established concept of user stories to achieve security requirements traceability and thus open the door to excellent security assurance, precisely because of their informal and lightweight nature.

1. Introduction

This paper proposes to extend agile practices to deal with security in an informal, communicative and assurance-driven spirit.

Agile processes are very effective at delivering systems customers want. It is not the object of this paper to establish the extent of this success, nor to provide an exhaustive list of contributing factors. Instead of reiterating the ground covered in [4] and discussing the broad range of agile practices as applied in security sensitive projects, this paper focuses on what is arguably the biggest obstacle to their adoption, namely, their lack of adequate tracking of security requirements. As a result of the inability of current agile practices to trace security requirements, agile security assurance may be viewed as inadequate. This is a shortcoming that is fairly easy to fix, however, by adding abuser stories to the agile developers' palette. It is not an inherent weakness as suggested in [10], for example, where agile processes are argued to be unsuitable for developing secure software due to the lack of formality in agile requirement specifications. There is nothing intrinsically virtuous in writing formal specifications. They are only desirable to the extent that they promote better assurance arguments. Despite their informality, agile methods excel in functional assurance. Indeed, test-driven development places assurance squarely at the heart of development. Moreover, assurance is better served by communication with stakeholders than by formality. Security's defining feature is the historical and continued standoff between defenders and attackers. No system is ever completely impervious to attack. Attackers invariably respond with novel attacks as engineers improve protection measures. A system

considered secure is in a state of unstable equilibrium; its ability to fend off attacks ceases as its environment changes, whether through technological breakthrough or by an increase in motivation or resources. To remain secure, the system must change as the environment changes, preferably by anticipating changes, certainly by embracing attackers' progress. These observations suggest that the demands of security engineering fit well with the agile mindset.

Section 2 summarizes the current agile approach to requirements engineering. While it is recognized in section 3 that current agile practices cannot deal with a range of security requirements, this can be easily rectified by extending traditional user stories with abuser stories as discussed in section 4. *Abuser stories* is a new concept, based on abuse cases. Abuse cases were first discussed in [8]. Section 5 contains advice on writing effective abuser stories.

2. Current Agile Requirements Engineering Practices

Ill-understood requirements are the most common cause of failure of software projects. In response, more formal techniques for expressing requirements unambiguously have been developed. However, agile methods shun formal requirements documents. Instead, they focus on continuous validation and feedback on work in progress.

The position of requirements is ambivalent in agile processes as relatively little effort is expended on them, yet development is driven by requirements as shown in paragraph 2.2.

2.1. User Stories

The role of user stories in agile development has been described extensively in the literature, notably in [1]. Summarizing, user stories are brief, informal descriptions of requirements written by the system's customers. They illustrate how the system can be used to create value.

User stories only provide enough detail to facilitate estimations of how long each story will take to implement with reasonable accuracy. Detailed requirements are given in face to face, informal meetings between the customer and developers at implementation time.

User stories express a capability of the system under development that delivers business value. They are

ranked according to their perceived value by the customer. They acquire a score according to their ranking. This score may vary throughout the software development life cycle as the market changes.

Effort estimates, on the other hand, are made by developers. Techniques for arriving at accurate estimates have been described in [3] and are fun to learn through the XP Game [9].

2.2.Planning

Agile development is iterative. This means that development is divided into short periods, referred to as *iterations*. The goal of each iteration is to realize the greatest possible value within the available time frame. At the end of each iteration, the system's customers check whether the system satisfies the requirements captured in the pertinent user stories by performing each user story's acceptance tests.

3. The Case for Extending Agile Practices

It is not unusual to find security-related details in user stories. Consider, for example, the following fragment of a user story for a web gambling application:

... The user fills in the amount of the stake and plays. ...

Development teams will soon realize that, in order to generate sustainable business value, the application must authenticate the user. The user story may therefore be rewritten thus:

... The user authenticates himself with a password. He fills in the amount of the stake and plays. ...

In this instance, a countermeasure to the security concern that the attacker may impersonate a user, can indeed be expressed in the user story. Whether this is advisable is debatable, but, in any case, some types of attacks cannot be so expressed. Consider, for example, the classic case described in [2] where Cigital engineers found vulnerabilities in the shuffling algorithm of an online poker game that allowed them to determine which cards their opponents had been dealt. In such cases, an extension to user stories is needed to describe the threat. Abuser stories are the extension proposed here. They are discussed in detail in the next section.

4. Enhancing Agile Requirements Engineering with Abuser Stories

4.1.Definition

Abuser stories identify how attackers may abuse the system and jeopardize stakeholders' assets. Thus they state systems' security requirements. Similar to user stories, they do so briefly and informally.

System abuse, or attack, carries a cost which amounts to negative business value.

Similar to ranking and scoring user stories according to business value, abuser stories may be ranked and scored according to the perceived threat they pose to customers' assets. The ranking must take into account both how much damage may be done and the likelihood of a successful attack. The score given to abuser stories should be commensurate with scores of user stories. In other words, user story value and abuser story cost should be equal if the abuser story is expected to wipe out the earnings from the user story.

User story value may change as market conditions change. Similarly, abuser story cost may change as the environment changes. A technological breakthrough, for example, may make an attack easier and therefore more likely. Assets may become more attractive targets. Adversaries may become better funded. Similar systems may since have been secured, making the system being developed the weakest in its class. These are all factors external to the project. But internal factors may also change the risk weighting of an abuser story. For example, countermeasures taken in previous iterations may increase the risk of an abuser story, because it has become the easiest way to attack the system.

Effort estimates are given for each abuser story as input to the planning game. Estimates cover the effort required to implement countermeasures to threats described in the abuser story.

Refutation is to abuser stories what acceptance testing is to user stories. Refutations address common vulnerabilities and typical attacks which may lead to an abuser story executing. They demonstrate that described attacks are impossible, or at least implausible. Indeed, risk never goes away completely as a system is never completely secure. However, risk must be shown to have been reduced to acceptable levels.

As assets are exposed through the functionality offered by user stories, abuser stories only become pertinent when at least one user story enabling the attack described has been implemented. The story of the players who know their opponents' hand, for example, only becomes pertinent when the poker game is added to the gambling site.

4.2.Planning

Implementing a user story increases the attack surface of a system and consequently the risk of abuse. Business value realized in an iteration must therefore be adjusted with the cost of absorbing risk created by user stories.

Introducing abuser stories allows business value to be tracked more accurately and facilitates rational planning of the effort required for security-related development. More particularly, considering user and abuser stories together ensures explicit and rational trade-off between functionality and security.

As risk mitigation reduces risk absorption costs, but requires effort, iteration plans for security-sensitive projects would not only include user stories that will be realized, but also abuser stories that will be refuted.

5. Writing Effective Abuser Stories

As has been stated, security requirements may be described briefly and informally as abuser stories. Although they are lightweight, low-effort and informal, they are sufficient to trace requirements. However, requirements traceability is but one of the necessary conditions for good security assurance. This section addresses two others, namely completeness and accuracy.

A set of abuser stories is effectively the skeleton of a threat model. Threat models have been discussed extensively. Particularly their treatment in [6] brought them to the attention of the development community. Many of the ideas discussed in this section stem from [1]. Its treatment of the discipline of security engineering is inextricably bound with threat models.

This section describes practices which aid writing high-quality abuser stories cost-effectively. Section 5.1 advises the involvement of as many people from diverse backgrounds as possible. Sections 5.2 and 5.3 examine some of the sources of inspiration for abuser stories.

5.1.Abuser Story Authors

User stories are written by customers. Customers should also be involved in writing abuser stories, as they are attuned to the business assets which need protection. However, to achieve a good threat coverage quickly, it is essential to draw on the expertise of developers, because many hands make light work and because developers' distinctive areas of expertise tend to make them sensitive to certain types of threats sooner than non-technical authors. Some of the system's assets are, by definition, of a technical nature. In the example of the gambling web site, it is likely that customers will quickly come up with threats to various accounts. For example, they may point out that accounts holding users' gains must be protected from attack. Threats to the randomness of the gambling

process, on the other hand, are more readily identified by a developer.

So abuser stories depart from traditional agile requirements engineering to the extent that they are not exclusively written by customers, but jointly with the development team. They reinforce the agile principle of involving all team members in a broad spectrum of activities. No-one is deemed to have a monopoly on a given area of expertise.

5.2.Assets

Assets are a good starting point for writing abuser stories. Anything of value to the customer which is potentially accessible through the system, should be considered a target. An asset may have intrinsic value, such as money in a bank account, or it may derive its value from its role in revenue generation, such as the card shuffling algorithm at a poker site. Since the online casino's appeal to gamblers is likely to diminish if the algorithm is perceived to be unfair, a fair algorithm should be considered as an asset. An unfair one is arguably a liability. Such assets are harder to identify, but will tend to show up when examining who the attackers are, their motivation, resources and expertise.

5.3.Attackers

The nature of an attack is largely determined by the kind of adversary. It therefore pays to reflect on who potential abusers may be. Pertinent factors include the resources they command, their skills, motivation and risk aversion.

Predators co-evolve with their prey and hence sensitivity to the species that inhabit the customer's ecosystem is required. The history of the customer's industry is typically a good guide to the motivation and even the attack techniques.

Skills and resources are, in a certain sense, interchangeable as a resourceful adversary can hire skillful mercenaries. Organized crime is a resourceful adversary. So are intelligence agencies or terrorists. However, their motivations are different and they will go after different targets, use different techniques and have a distinctive risk assessment.

Attackers are unlikely to invest many resources unless they have a clear motive. At the other end of the spectrum lie low-investment acts of vandalism.

Threats from low-skilled system users may have devastating consequences. Secret gamblers using the example gambling site may rather deny using the site than settle their debts.

Customer staff are a rich source of inspiration for potential attackers. The majority of fraud cases occur with inside help.

6. Further Work

Abuser stories have served me well in a number of assignments. However, they were used surreptitiously without explicit management recognition of their role in tracing security requirements. In this sense, the concept has been proven, but the implementation is missing.

While the current lack of requirement traceability is, in my view, the greatest obstacle to providing security assurance in agile processes, considerable work certainly remains to be done on assurance as was pointed out in [5]. Refutations as described in [7] may prove to be a good foundation.

7. Conclusions

Abuser stories are a non-invasive extension to agile practices providing security requirements traceability.

8. References

- [1] Anderson, R. J., *Security Engineering. A Guide to Building Dependable Distributable Systems*, Wiley, 2001
- [2] Arkin, B., et al, "How we Learned to Cheat in Online Poker: A Study in Software Security", 1999, available at http://www.cigital.com/papers/abstracts/developer_gambling.html
- [3] Beck, K., *Extreme Programming Explained*, Addison-Wesley, 1999 (1st ed.) and 2004 (2nd ed.)
- [4] Beznosov, K., "Extreme Security Engineering: On Employing XP Practices to Achieve 'Good Enough Security' without Defining It", *First ACM Workshop on Business Driven Security Engineering (BizSec)*, Fairfax, VA, 31 October, 2003, also available at <http://konstantin.beznosov.net/professional>
- [5] Beznosov, K., and P. Kruchten, "Towards Agile Security Assurance", *Proceedings of The New Security Paradigms Workshop*, White Point Beach Resort, Nova Scotia, Canada, 20-23 September 2004, also available at <http://konstantin.beznosov.net/professional>
- [6] Howard, M., and D. LeBlanc, *Writing Secure Code*, Microsoft Press, Redmond, 2003
- [7] McDermott, J., "Abuse-Case-Based Assurance Arguments", *Proceedings of the Annual Computer Security Applications Conference*, ACSA Publications, Silver Spring, 2001, also available at <http://www.acsac.org>
- [8] McDermott, J., and C. Fox, "Using Abuse Case Model for Security Requirements Analysis", *Proceedings of the Annual Computer Security Applications Conference*, ACSA Publications, Silver Spring, 1999, also available at <http://www.acsac.org>
- [9] Van Cauwenberghe, P., and V. Peeters, *The XP Game*, <http://www.xp.be/xpgame/download.html>
- [10] Viega, J., and G. McGraw, *Building Secure Software*, Addison-Wesley, 2002