Penetration Tests a Turning Point in Security Practices? Organizational Challenges and Implications in a Software Development Team[*]

Sven Türpe
Fraunhofer SIT
64295 Darmstadt, Germany
tuerpe@sit.fraunhofer.de

Laura Kocksch
Fraunhofer SIT
64295 Darmstadt, Germany
kocksch@sit.fraunhofer.de

Andreas Poller
Fraunhofer SIT
64295 Darmstadt, Germany
poller@sit.fraunhofer.de

ABSTRACT

Many software vendors conduct or commission penetration testing of their products. In a penetration test security experts identify entry points for attacks in a software product. The audits can be an eye-opener for development teams: they realize that security requires much more attention. However, it is unclear what lasting benefits developers can reap from penetration tests. We report from a one-year study of a penetration test and its aftermath at a major software vendor, and ask how an agile development team managed to incorporate the test findings. Results suggest that penetration tests improve developers' security awareness, but long-lasting change of development practices is hampered if security is not properly reflected in the communicative and collaborative structures of the organization, e.g. by a dedicated stakeholder. Based on our findings we suggest improvements to current penetration test consultancies by addressing communication and organizational factors in software development.

Keywords

Development practices; secure software engineering; penetration testing; organizational factors; structure-agency-duality; qualitative study

1. INTRODUCTION

Software security is in demand and the industry applies numerous practices of security development [5]. Most work on software security focuses on technology, but human and organizational factors are equally important to understand and support practitioners [10]. For example, social factors can influence the adoption of security tools [11]. Research on human-centered security [8] takes such factors into account but has so far focused on end users and their interaction with security mechanisms. Commercial software development, in contrast, is creative work by teams of trained professionals embedded in organizations.

We discuss early results of a study on software penetration testing. We observed a mature, yet security-inexperienced agile software development team during and after a penetration test of their product, followed by a training workshop. External security consultants hired by the team's employer, a major software vendor, spent two months attempting to hack an instance of the software and reviewing code to find vulnerabilities. The consultants then submitted their findings as tickets to the developers' issue tracker and later ran a three-day security workshop with members of the development team. We followed the development team for one year starting at the time of the penetration test.

Initially we aimed to study possible mid- and long-term effects of penetration testing on development practices. After we realized that there were little observable changes or improvements to development practices, we amended our research question and explored the challenges that practitioners on all levels – from individual developers to product management – encountered when transferring security insight from penetration testing and training into daily practice. Finally, we suggest to take organizational and structural factors into account for a longer lasting impact of security consultancies in a professional software development team. An aspect of further investigation that is only touched upon here is the relation between agile development principles and security as a structural and practical challenge.

Penetration tests and developer training are common industry practices [5]. While the immediate results of a penetration test are easy to measure, e.g. in the number of fixed security issues, the benefits for mid- to long-term security practices are less clear, e.g. organizational initiatives for proactive security work.

Penetration tests [4] are costly as they require human labor, but they can be a profitable security investment [2]. Security experts explore a system or program from an attacker's perspective to find vulnerabilities. The result is a list of security flaws that should be read as a sample of the vulnerabilities present rather than as a definitive list of defects to fix. Yet it seems that developers often go for the quick win of fixing the issues on the list without analyzing root causes and making more profound changes [4]. What can we learn about the reasons for their actions? The developers in our setting follow agile practices that are common in the software industry. Our study thus also promises some insight into the question of agile security engineering and assurance [1]. Finally, security training obviously aims for long-term effects. In security training, teaching attacks and exploits is common and believed to be beneficial. Does this have lasting effects beyond awareness? Might developers have further needs that this approach does not fulfill?

The HCI literature often describes organizations as frameworks for technological and human interaction. Orlikowski [6]

[*]This is a revised version of a CHI'16 poster abstract [7].

shows that technologies entering organizations have reciprocal effects and co-constitute each other. She takes into account that technologies are sustained of and reaffirmed by human action, but also that practices only occur in the context of the organization´s structure. This takes upon the theory that structure and agency are co-constituted. Organizations therefore exist in a duality of structure and agency. Suchman [9] points out in regard to the design of technologies that there is a difference between plans and actions. Plans consider a goal, whereas actions are situated and are necessarily vague to react on obstacles on the way. We expect in line with these accounts that implementing change in the daily practices of developers is closely related to the organization's structure.

2. METHODOLOGY

Our field site is an agile software development project at a multinational enterprise comprising 37 team members in seven Scrum teams spread over premises in Asia, Europe and North America. A contracted consultancy firm executed a two-months penetration test of the product, a web dashboard to analyze business process indicators. The consultants also performed a final on-site training workshop with 23 of the 37 product team members one month after they handed over the test results. We accompanied the product team for one year, for a period comprising one full and two half product release cycles, see Figure 1.

Questionnaires. At the beginning, we created two questionnaires: the first, before test results were reported, asked team members about current secure software development practices, their expectations of the penetration test, and a self-assessment of their security knowledge. The second questionnaire, before the training workshop, asked how developers had attended to identified security issues so far and what they expected of the upcoming workshop. Questionnaires had 17 and 25 questions, free text fields and Likert-scale items. Response rates were 42% and 33%.

Workshop Observation. Two researchers observed the subsequent workshop. They took notes in individual field diaries to report the actions of workshop participants and of the consultant who acted as an instructor. The researchers focused on situations where both parties interacted. In particular, they aimed to observe explicit or implicit negotiations of how security should be treated in software development and of who was to be responsible for taking on security-related tasks given their particular role in the team.

Data Analysis and Document Review. In the two months after the workshop, the team was heavily engaged in fixing the found security weaknesses, almost all of which could be resolved. We followed this process by analyzing data from the team's internal issue tracking system and an internal wiki to understand work flows and communication patterns. We also monitored wiki entries and minutes from coordination meetings on the topic of security.

Interviews. Six to ten months after the workshop we interviewed 14 product team members – developers and managers – and the consultant. Our semi-structured interviews aimed to find out how fixing actions spread among the different Scrum teams, and which follow-up activities and changes in development practices teams implemented to prevent further similar defects. After we found that practices had changed little, we focused on learning more about the interviewees' roles in the organization and how their involvement in security reflected this role. We gained insight into how security was framed as a requirement in development before and after the penetration test, what aspects of developers' work routines remained unaffected and its reasons. Interview data and field diaries were transcribed and coded. We derived codes both from the interview guideline and from key themes found in the data, e.g. issue fixing practices, communication practices, management and developer perspective. The coding was conducted by two individual researchers. Other material (i.e. wiki, ticket system) was not coded but only used to prepare interviews or improve our understanding of the setting.

3. PRELIMINARY RESULTS

3.1 Questionnaires

15 out of 37 team members responded to our first questionnaire. Their work experience varied considerably: four participants had less than five, seven between five and ten, and four more than ten years of work experience. Participants' descriptions of their work area included various development topics, roles and activities. They reported to have good general knowledge in software development from "experienced' to "expert", but very little knowledge in security. Only one participant considered himself a security "expert". As their expectation for the security audit, eight out of 15 developers stated a high interest in the results, seven responded "neutral" and nobody stated a lack of interest. Most respondents agreed or strongly agreed that they hoped the audit to be an impulse to start continuously improving the product security; three responded "neutral".

Twelve team members responded to our second questionnaire which we distributed after the hand-over of the audit results but before the security workshop. Out of those twelve, eight had not reviewed the test results before the workshop. Only one participant stated that his team had discussed the test results, but ten out of twelve participants were excited about the upcoming workshop.

To summarize, the team had rather diverse work areas and professional experience. Team members considered their general development skills as advanced, but software security knowledge as fairly limited. Respondents were mostly curious about the penetration test and had high expectations of the upcoming workshop. A majority wished for profound changes of their team's security practices.

3.2 Workshop Observation

The workshop had three parts: first, a talk to raise awareness of security as an important topic for developers, second, hands-on hacking exercises, and third, a group hacking challenge. We witnessed a very productive workshop; developers seemed focused and enjoyed the work. The consultant appeared eloquent, used persuasive rhetoric and showed much attention to detail and technical expertise. He was very engaged and aimed to get every participant involved. He seemed to fulfill the role of a stakeholder for security by communicating requirements, viewpoints, rationales and calls for actions for a more security-aware product development. His strong rhetoric supported this impression: he often switched to dichotomous language, security is either "there" or "not", developers must "trust no one", and actors are either "good" or "evil".

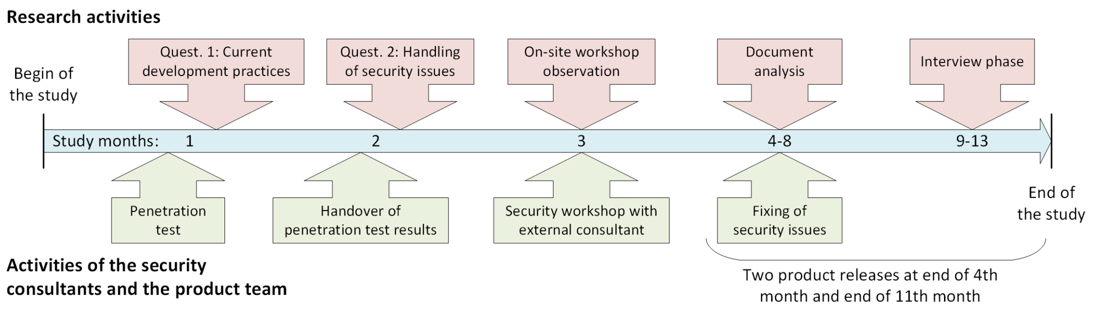
Research activities
Quest. 1: Current development practices
Quest. 2: Handling of security issues
On-site workshop observation
Document analysis
Interview phase
Begin of the study
Study months: 1 2 3 4-8 9-13
End of the study
Penetration test
Handover of penetration test results
Security workshop with external consultant
Fixing of security issues
Activities of the security consultants and the product team
Two product releases at end of 4th month and end of 11th month

Figure 1: Timeline showing activities of researchers and external consultants / development team

However, the workshop was more problem-oriented than solution-oriented. The consultant was strong at conveying knowledge about security flaws and pitfalls, but he rarely participated in discussions about how to actually solve revealed security issues, and how to implement further security-related activities. The workshop left open how to address security in a development team context and the company's ecosystem, e.g., by establishing particular security roles.

3.3 Data Analysis and Document Review

After the workshop, we analyzed the content of the team's issue tracking system and meeting minutes in their internal wiki. The consultant had reported 56 vulnerabilities, 24 of them rated *critical* and 14 *high* on a four-step severity scale. Issue types included server- and client-side code injection vulnerabilities, broken cryptography, information exposure at the application's interface, as well as access control flaws. Overall the consultant used 22 different CWE IDs (see http://cwe.mitre.org/) to classify the findings.

The majority of issues (about 77%) were fixed in the two months after the workshop; only 4% were solved before the workshop. Whether this was due to the workshop or to the release cycle remains unclear. The developers worked for more than 4 months on the remaining issues.

Meeting minutes recorded discussion on how to coordinate fixing the identified issues, but we did not see any further discussion about security or transfer of the findings to other areas. One Scrum team added a few generic security test cases to their readiness checklist, such as "think about whether the feature may have introduced new security vulnerabilities", but no more specific actions were documented. We found no additional security content in the wiki, which the team used intensively for preparing and documenting work tasks, meetings, and technical matters.

3.4 Interviews

We executed semi-structured interviews six to ten months after the security workshop. None of the interviewed developers considered themselves to be a "security expert" after the workshop. Neither could they point at a security expert in their team or other persons responsible for the product's security. One interviewee answered: "Good question. All of us [are responsible], or not?".

Changes After Consulting. Many developers thought the main outcome of the penetration test and the workshop to be "awareness", e.g., one developer stated "I feel like I have a lot greater awareness of the subject matter". Some interviewees described this more specifically as a change in perception of security risks. For example, one interviewee pointed out: "no matter what you do, somebody is trying to penetrate what you do." Some described the workshop as an "eye opener" for realizing that security matters for their work. However, they linked awareness exclusively to other developers, mostly only within their own Scrum team. The management was never mentioned as a security stakeholder, but as a stakeholder for feature development and the development schedule.

While participants highlighted awareness as the major outcome they could not point to any specific changes in their work routines. When asked for the biggest changes in their work process one participant answered: "The biggest changes? They were not that big. [...] I don't know a good answer to that." Interviewees also said the development team had not created additional artifacts as reminders or for knowledge transfer while fixing the security issues: "What kind of artifacts? [repeating the interviewer's question] I cannot think of any." This finding pointed us to a lack of coordination and communication about security beyond fixing the issues revealed by the security audit.

No Additional Resources. One reason for this lack of attempts to further anchor security as a development topic is that feature development received higher priorities than improving the product security. One developer stated "[w]hat they [the developers] are considering is [...] if security is not on the list, then is it really worth the time and extra energy to do it?" pointing out that developers did not receive additional resources for security. The developers were thus caught in a situation where they were committed to producing a high-quality product without security flaws, but were limited to personal efforts or low-key agreements within their own Scrum team. Security was not considered a sales pitch for the management, hence not requested by customers and not on the feature list.

Not only was security not on the list of feature development, developers felt their superiors would not give them credit for security initiatives: "if we did not do it, nobody would notice". Apparently, there was no structure helping developers to act upon security. One participant reflected on that and said: "it would be better if it was clarified [as a requirement] from above". Feature development does not just come first and has a higher priority but is the only context where communication between developers and management worked. During the security issue fixing there were no clear guidelines who, when or how they should be fixed. The different product teams resolved the issues highly idiosyncratically and with only informal communication between them. No knowledge distribution was organized and several developers stated they were eager to get to know something about the fixing process, especially on their own code, even though they were busy with feature development.

Setting Priorities. Fixing the issues was considered differently by the developers and the management. While management was concerned with getting the "number down" so they could "argue internally", developers had in mind risks for the product's release. Particularly "big" issues were those that had many cross-component dependencies and thus, when fixed locally, jeopardized the product's functionality. Security in terms of the security issues was translated into the fixing process, but no equivalent was found after the fixing was completed. Security had not entered daily practices nor its structural surroundings. Ensuring the product's release and developing new features were the goals that management and developers agreed upon.

The process of setting development priorities did not become clearer during further interviews. Interviewees from management considered security as a pervasive quality of the product, hence, understood security as an issue autonomous Scrum teams needed to address in their development activities. They deemed security training sufficient to foster this understanding. But since security as a product feature was supposedly invisible to customers, it was not a topic further discussed between development and product management.

4. DISCUSSION AND CONCLUSION

We found the security consulting successful yet limited in its effect. Penetration testing and training were successful in the sense that interviewees enjoyed the workshop, reported increased security awareness, and were able to address the issues reported to them. However, our participants did not report tangible changes in their development practices or planned product modifications beyond fixing the reported vulnerabilities. What prevents them from going further and where might they need support?

Adopting secure development practices is not simply a developer awareness problem, but requires dealing with complex organizational and social factors in software developing companies. This is in line with Orlikowski´s theme of structure and agency in co-constitution [6]. Development work is organized around the exchange of goals and results between management and Scrum teams; quality goals are only communicated as ensuring the product's release and its feature development. Following the Scrum methodology, development teams are self-organizing and hold all technical expertise needed, whereas management aligns requirements with business needs. Reinforced by actions of both parties, this structure resists change like adopting security practices.

Developers expressed needs for knowledge distribution and security requirement definitions, but the organizational structure hampers efforts to satisfy these needs. Product management prioritizes features by expected business value. Security is deemed not marketable as a feature and treated merely as a matter of quality to be addressed by developers. It is nobody's role to counteract this bias since a "security buddy" [3] or similar roles do not exist. On the other hand, management encourages defect fixing by monitoring defect counts and enforcing policies.

Facilitating learning beyond awareness and defect fixing may however require more than only adjusted management priorities. Our workshop observations highlight a gap. While vulnerabilities and attacks are an important element of security training, developers and their managers ultimately need coping strategies, such as design patterns, development practices, and aids for decision-making. While the agile idea of self-organizing teams can facilitate experimenting with practices and tools, someone has to invent and supply those.

Mentioned organizational and structural factors could be addressed by activities and tools, e.g., training could increase managers' awareness of organizational obstacles to security. Tools could make the product's security status visible to product management fostering an understanding that security can be a feature, can become visible and can be marketed to customers. Security audits could also feature suggestions for how to foster discussions of security's role in the organization. If we want to incorporate security in organizations we have to take into account social factors like the relationship between structure and practice.

From viewpoints that specifically consider communication across distributed teams our findings may not come as much of a surprise. But in secure software engineering organizational and collaborative factors of development are underappreciated. This field would thus profit from further attention from HCI and CSCW scholars and an organizational perspective on security requirements.

5. REFERENCES

[1] K. Beznosov and P. Kruchten. Towards agile security assurance. In *Proc. New Security Paradigms Workshop*, NSPW '04, pages 47–54. ACM, 2004.

[2] R. Böhme and M. Félegyházi. Optimal information security investment with penetration testing. In T. Alpcan, L. Buttyán, and J. S. Baras, editors, *Proc. GameSec '10*, pages 21–37. Springer, 2010.

[3] S. Lipner. The trustworthy computing security development lifecycle. In *Proc. ACSAC'04*, pages 2–13, Dec. 2004.

[4] G. McGraw. *Software Security: Building Security In.* Addison-Wesley Professional, 2006.

[5] G. McGraw, S. Migues, and J. West. Building Security In Maturity Model (BSIMM) version 6, 2015.

[6] W. J. Orlikowski. The duality of technology: Rethinking the concept of technology in organizations. *Organization Science*, 3(3):398–427, 1992.

[7] A. Poller, L. Kocksch, K. Kinder-Kurlanda, and F. A. Epp. First-time security audits as a turning point? challenges for security practices in an industry software development team. In *Proc. CHI'16 EA*, pages 1288–1294. ACM, 2016.

[8] A. Sasse. Designing for Homer Simpson-d'oh. *Interfaces: The Quarterly Magazine of the BCS Interaction Group*, 86:5–7, 2011.

[9] L. A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication.* Cambridge Univ. Press, 1987.

[10] R. Werlinger, K. Hawkey, D. Botta, and K. Beznosov. Security practitioners in context: Their activities and interactions with other stakeholders within organizations. *Int. J. of Human-Computer Studies*, 67(7):584–606, 2009.

[11] S. Xiao, J. Witschey, and E. Murphy-Hill. Social influences on secure development tool adoption: Why security tools spread. In *Proc. CSCW '14*, pages 1095–1106. ACM, 2014.