First-time Security Audits as a Turning Point? Challenges for Security Practices in an Industry Software Development Team

Andreas Poller
Fraunhofer Institute for Secure Information Technology
64295 Darmstadt, Germany
poller@sit.fraunhofer.de

Laura Kocksch
Fraunhofer Institute for Secure Information Technology
64295 Darmstadt, Germany
kocksch@sit.fraunhofer.de

Katharina Kinder-Kurlanda
GESIS - Leibniz Institute for the Social Sciences
50667 Cologne, Germany
katharina.kinder-kurlanda@gesis.org

Felix Epp
Fraunhofer Institute for Secure Information Technology
64295 Darmstadt, Germany
fepp@sit.fraunhofer.de

CHI'16 Extended Abstracts, May 07-12, 2016, San Jose, CA, USA
ACM 978-1-4503-4082-3/16/05.
http://dx.doi.org/10.1145/2851581.2892392

Abstract
Software development is often accompanied by security audits such as penetration tests, usually performed on behalf of the software vendor. In penetration tests security experts identify entry points for attacks in a software product. Many development teams undergo such audits for the first time if their product is attacked or faces new security concerns. The audits often serve as an eye-opener for development teams: they realize that security requires much more attention. However, there is a lack of clarity with regard to what lasting benefits developers can reap from penetration tests. We report from a one-year study of a penetration test run at a major software vendor, and describe how a software development team managed to incorporate the test findings. Results suggest that penetration tests improve developers' security awareness, but that long-lasting enhancements of development practices are hampered by a lack of dedicated security stakeholders and if security is not properly reflected in the communicative and collaborative structures of the organization.

Author Keywords
Development practices; secure software engineering; penetration testing; organizational factors; qualitative study

ACM Classification Keywords
D.2.m [Software Engineering]: Miscellaneous

Penetration testing in software development
Penetration tests are evaluations of network hosts, whole networks, or specific software applications that aim to find weaknesses that may compromise security goals [2]. Often testers "penetrate" running IT systems directly, but the evaluation can also involve, e.g., source code analyses. Penetration testing is mostly done manually, but heavy use of supporting tools is common.

Studied case
External consultants were contracted (a) to set up a test environment, (b) to identify vulnerabilities in source code and at a running instance of the software product, (c) to document findings, and (d) to perform a security workshop. They reported 56 issues; 24 of critical severity. 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.

Introduction

Investigating software development practices and how software tools can support communication and collaboration in software-developing enterprises has repeatedly been a topic for HCI and CSCW scholars. Some recent studies focused specifically on IT security and suggested possible improvements for tools and processes in secure software development activities. Werlinger et al. showed that human, organizational, and technological factors are equally important in order to understand and support security practitioners [6]. They found the design of security tools such as intrusion detection systems to lack this support as it does not take into account the complex environment and multi-faceted ecosystems in which security practitioners interact.

Xiao et al. described the way in which social factors considerably influence security tool adoption. They developed advice both for organizations that want to employ security tools, and for the developers of these tools [7]. They also highlight that the "preventive nature" of security activities and tools, aimed at *preventing* events in the future, makes their benefits less tangible and therefore imposes challenges for establishing a security culture in companies. However, while over the last decade much research in HCI has focused on making the *design* of security mechanisms more user-friendly and human-centered [5], the people in charge of engineering and operating security-relevant software have received not nearly as much attention.

We present early results of a longitudinal study focusing on a common turning point in the engagement of software developers with security topics: first-time security audits. We investigated a mature, yet security-inexperienced Agile software development team during and after their product underwent a thorough *penetration test* for the very first time. After new internal requirements had been elicited for the product, the team's employer, a major software vendor, decided to hire external security consultants to audit the software. For two months the consultants attempted to "hack" a running instance of the software, reviewed its source code and finally reported the test results in a written report and by submitting tickets to an internal trouble ticket system. In addition, the consultants performed a three-day security workshop with members of the affected product team.

According to industry studies, such penetration tests are among the top five secure development practices in major software-developing enterprises [4]. Across the industry, first-time security audits repeatedly take place for well-established development teams that either experience sudden, unexpected security incidents, or face new challenges to deal with security risks when product application scenarios evolve (e.g. a software formerly used in self-contained networks becomes a publicly accessible service). In our scenario the product was a new acquisition that needed to be tested for cross product dependency issues in particular.

We followed the development team for one year starting at the time of the penetration test. Initially the vendor asked us to study possible mid- and long-term effects of penetration testing on development practices to find arguments for justifying the considerable costs of these audits. While [1] argue that penetration testing can be a profitable security investment, the benefits for mid- to long-term security practices are in fact less clear. The vendor was thus interested to learn how penetration tests could foster an understanding among developers to integrate security in all development life cycle phases. 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 the product management - encountered

Data collected in the study

Questionnaires
We distributed questionnaires before security issues were identified and reported, and before the security workshop took place (response rates 42% and 33%).

Observational data
Two researchers observed the three-day security workshop. They recorded sequences of interactions enriched with their own impressions in field diaries.

Internal documents
Security issue were reviewed regarding type, work progress, developers involved, actions taken and related discussions. Using the internal Wiki each interviewee was screened for her work area, as well as tasks and roles in the project.

Interviews
We conducted 15 semi-structured one-on-one interviews (14 hours overall), 5 with video calls, 10 in person.

when transferring security practices learned from penetration testing activities into daily workflows.

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 with the product which was 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. At the beginning, we created two questionnaires: the first, distributed 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, distributed before the training workshop, asked how developers had attended to identified security issues so far and what they expected of the upcoming workshop. Questionnaires consisted of 17 and 25 questions, free text fields and Likert-scale items.

The subsequent workshop was observed by two researchers. They took notes in individual field diaries to report the actions of product team members attending the workshop, 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.

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 software development management system and an internal Wiki system to gain insights into work flows and communication patterns.

Six to ten months after the workshop we conducted semi-structured interviews with 14 product team members (developers and managers) and the external consultant. In the interviews we 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 changed little, we focused on learning more about interviewees' roles within the organization and how this role was reflected in their engagement in security topics. We gained insights 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 the reasons for it.

Interview data and field diaries were transcribed and coded. We derived codes both from the interview guide and from key themes found in the data. Other material (i.e. wiki, ticket system) was not coded but only used to prepare interviews or improve our understanding of the setting.

Preliminary Results

Questionnaire results
15 out of 37 team members responded to our first questionnaire. We found work experience to vary 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, see Figure 2.

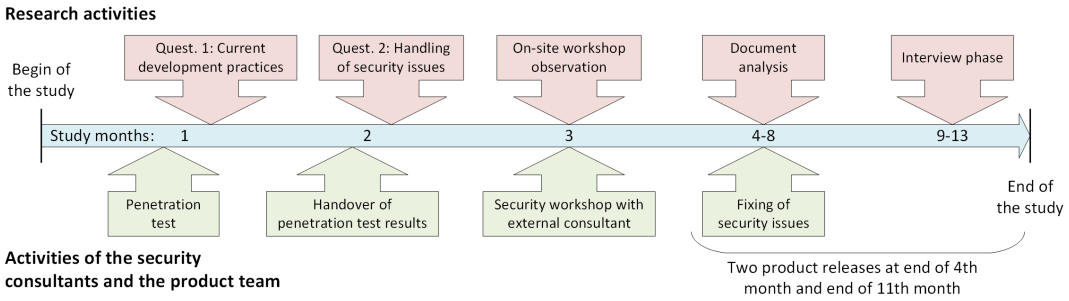

Figure 1: Timeline showing activities of researchers and external consultants / development team

With regard to software development knowledge, eight participants considered themselves to be “experienced” and seven even as “experts”. In contrast, regarding security knowledge, two participants considered themselves as “novice”, twelve as “advanced beginners” and only one as an “expert”. With respect to expectations of the security audit, eight out of 15 developers stated a high interest in the results, seven responded with “neutral” and nobody stated a lack of interest. A vast majority of respondents agreed or strongly agreed that they hoped the audit to be an impulse to start continuously improving the product security; three responded with “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. The majority (eight participants) had not reviewed the identified security issues before the workshop. Nevertheless, ten out of twelve participants were excited about the upcoming workshop.

To summarize, the team was relatively diverse with regard to 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 fundamental changes of security practices within their team.

Workshop observation

The workshop was split into 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 the workshops to be very productive; developers seemed focused and enjoyed the work. The consultant was seen to be 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 security stakeholder 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”.

Figure 2: Word cloud calculated from participants' descriptions of their own work and role in the team (font size reflects number of mentions)

However, the workshop was more problem-oriented than solution-oriented. The consultant was strong at conveying knowledge about common 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 empower developers to integrate matters of security in a development team context and the company's ecosystem, e.g., by establishing particular security-related roles.

Document analysis

After the workshop, we analyzed the team's issue tracking system. We found that submitted security issues comprised a variety of security weakness types and ratings. The majority of issues (about 77%) were fixed in the two months after the workshop; only 4% were solved before the workshop. The developers worked for more than 4 months on the remaining issues.

We analyzed meeting minutes stored in an internal Wiki and found that while it was discussed how to coordinate fixing the identified issues, no further discussion about security or transfer of the findings to other areas occurred. A few generic security test cases such as "think about whether the feature may have introduced new security vulnerabilities" were added to readiness checklists, but no more specific actions or results were documented. We found no additional security content in the Wiki although the team intensively used it for preparing and documenting work tasks, meetings, and technical matters.

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 security 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?".

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 interviewees 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.

While participants highlighted awareness as the major outcome they could not point to any concrete 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 described the development team not to have created additional artifacts as reminders or for knowledge transfer while they fixed 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.

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.

The process of setting priorities in development did not become clearer during our 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 incorporate in their development activities. They concluded that security training was sufficient to foster this understanding. On the other hand development managers also acknowledged the limited resources for security, but their decisions where to spend time were strongly influenced by "strategic targets". These targets were considerably shaped by a distinct product management. But as security as product feature was supposedly invisible to customers, it was not a topic discussed between development and product management.

Discussion and Conclusion

During our study we found that although the first-time security audit was successful, and interviewees confirmed that the security workshop was enjoyable and raised their awareness of security risks, our participants did not report tangible changes in development practices. Once we noticed this phenomenon we changed the focus of our study and tried to find out why it occurred.

We found several preliminary explanations we want to further investigate: first, the team was globally distributed, single development groups were specialized on individual feature areas and had different backgrounds. This caused a first gap between the individual groups and the management: teams enjoyed autonomy but as a consequence management attributed security to them as an intrinsic quality requirement. A similar split occurred on the management level: development management and product management were both specialized, and working priorities emerged from a cooperation of both parties. However, security had not been a topic in this cooperation so far, as neither side had addressed it. Product management perceived security as invisible to customers hence not marketable, and development management did not highlight security as something that could serve as a product feature. Both parties focused on short-to-market feature development; security could not (yet) serve this purpose.

Our results suggest that the incorporation of security activities such as first-time audits needs to focus not only on the individual skills of developers (Xie et al. show an approach for a different case [8]), but also on the overall setting of the affected project team. We revealed organizational and communication challenges to show that adopting secure development practices is not just a simple developer awareness problem, but requires dealing with complex organizational and social factors in software developing companies.

These 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.

From viewpoints that specifically consider communication across distributed teams our findings may not come as much of a surprise. For example, Matthiesen et al. have shown that cultural blind spots may hamper collaboration in distributed development teams [3]. But in secure software engineering organizational and collaborative factors of development are mostly unappreciated. This field would thus profit from further attention from HCI and CSCW scholars.

REFERENCES

1. Rainer Böhme and Márk Félegyházi. 2010. *Proc. GameSec '10*. Springer Berlin Heidelberg, Chapter Optimal Information Security Investment with Penetration Testing, 21–37.
2. Daniel Geer and John Harthorne. 2002. Penetration testing: a duet. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*. 185–195.
3. Stina Matthiesen, Pernille Bjørn, and Lise Møller Petersen. 2014. "Figure out How to Code with the Hands of Others": Recognizing Cultural Blind Spots in Global Software Development. In *Proc. CSCW'14*. ACM, New York, NY, USA, 1107–1119.
4. Gary McGraw, Sammy Migues, and Jacob West. 2015. *Building Security In Maturity Model (BSIMM) Version 6*. Technical Report. Cigital, Inc.
5. Angela Sasse. 2011. Designing for Homer Simpson-D'Oh. *Interfaces: The Quarterly Magazine of the BCS Interaction Group* 86 (2011), 5–7.
6. Rodrigo Werlinger, Kirstie Hawkey, David Botta, and Konstantin Beznosov. 2009. Security practitioners in context: Their activities and interactions with other stakeholders within organizations. *International Journal of Human-Computer Studies* 67, 7 (2009), 584 – 606.
7. Shundan Xiao, Jim Witschey, and Emerson Murphy-Hill. 2014. Social Influences on Secure Development Tool Adoption: Why Security Tools Spread. In *Proc. CSCW '14*. ACM, New York, NY, USA, 1095–1106.
8. Jing Xie, Heather Lipford, and Bei-Tseng Chu. 2012. Evaluating Interactive Support for Secure Programming. In *Proc. CHI '12*. ACM, New York, NY, USA, 2707–2716.