Surveying Secure Software Development Practices in Finland

Kalle Rindell
University of Turku
Turku, Finland
kalle.rindell@utu.fi

Jukka Ruohonen
University of Turku
Turku, Finland
jukka.ruohonen@utu.fi

Sami Hyrynsalmi
Tampere University of Technology
Pori, Finland
sami.hyrynsalmi@tut.fi

ABSTRACT

Combining security engineering and software engineering is shaping the software development processes and shifting the emphasis of information security from the operation environment into the main information asset: the software itself. To protect software and data assets, software development is subjected to an increasing amount of external regulation and organizational security requirements. To fulfill these requirements, the practitioners producing secure software have plenty of models, guidelines, standards and security instructions to follow, but very little scientific knowledge about effectiveness of the security they take.

In this paper, we present the current state of security engineering surveys and present results from our industrial survey ($n = 62$) performed in early 2018. The survey was conducted among selected software and security professionals employed by a selected set of 303 Finnish software companies. Results are compared to a commercial survey, the BSIMM version 8 and the similarities and distinct differences are discussed. Also, an analysis of the composition of security development life cycle models is presented, suggesting regulation to be the driving force behind security engineering in software industry.

KEYWORDS

software engineering, agile, security engineering, survey

ACM Reference Format:
Kalle Rindell, Jukka Ruohonen, and Sami Hyrynsalmi. 2018. Surveying Secure Software Development Practices in Finland. In *ARES 2018: International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3230833.3233274

1 INTRODUCTION

Information security and privacy of personally identifiable information are main concerns of all organizations involved with processing or storing digital data. *Security engineering*—the systematic effort to produce more secure computing environments [1]—is primarily concerned with the protection of operating environments and the post-development phases in the software life cycles. The overlapping of security engineering with software engineering has been nominal and more or less an afterthought. Security engineering covers external reviews, pre-scheduled milestones, and formal certifications, among other related processes. By and large, these do not contribute to the primary business requirements of software.

Several maturity frameworks, software life cycle models, and security standards have been established. The goal of these is to guide and commensurate software development processes towards producing secure software systems that meet formal security requirements with adequate security assurance. These include the Open Web Alliance Security Project's (OWASP) Software Assurance Maturity Model (SAMM) [17], the Building Security in Maturity Model (BSIMM) [13, 26], a generic design-oriented life cycle model from Synopsys, and the Security Development Lifecycle (SDL) from Microsoft [6, 14]. Of these, only BSIMM claims to be based on empirical observations by being created by observing and analyzing real-world data from leading software security initiatives.

There is a lack of scholarly empirical research on BSIMM, and to the best of our knowledge, this paper is the first to examine BSIMM from a scholarly viewpoint. This study's research objective is to evaluate BSIMM and Microsoft SDL against the security and software engineering practices currently used in the Finnish software industry, and contributes to research on the linkages between software and security engineering.

To this end, we conducted a survey among software professionals in selected software companies in Finland and asked for their security practises. The survey was conducted by presenting a comprehensive list of software security engineering activities to the survey respondents. The respondents were asked to rate these based on their usage, and to estimate the perceived effectiveness of these activities, and their impact upon security. The rest of the paper is structured as follows: section 2 briefly reviews the related work. Results are presented in Section 3 and discussed in Section 4. Conclusions follow in the final Section 5.

2 BACKGROUND

2.1 Secure Software Engineering

Software engineering has primarily been concerned with the effectiveness of software development and the means to effectively create features and functionality that fulfill the software's primary 'business' objectives [7, 8]. At the risk of over-generalization, in the grand scheme of things, security has been traditionally seen as a secondary objective that falls to the parent field of computer science [3]. Traditionally security engineering was seen as a part of system engineering, a more technical and non-theoretical subject primarily concerned with the task of protecting software systems from security threats. If software security was recognized as an explicit goal, it was seen as a subdivision of software quality improvement efforts. The actual software security was introduced after a software product had already been released and deployed. The

ARES 2018, August 27–30, 2018, Hamburg, Germany

ACM ISBN 978-1-4503-6448-5/18/08…$15.00
https://doi.org/10.1145/3230833.3233274

primary goal was to protect the operating environment, whether computers or networks, by guaranteeing confidentiality, integrity, and availability.

Introduction of maturity models, such as ISO/IEC SSE-CMM [9], in the 1990s, formalized a systematic approach towards security integrated into the software development processes. However, since the introduction of these quality and security improvement efforts, software development processes have undergone a comprehensive paradigm change. Managing software project and its risks is no longer based on careful pre-planning and evaluation (and likely rejection) of change requests, but rather adapting to the change [2]. Another prominent trend in software development methodologies is concentration on the functionality, or 'value'. In this frame of thinking, security requirements are prone to gaining less attention, or even being dismissed as secondary. In these agile and lean terms, anything that is not recognized as producing value, is treated as *waste* [18]. Tendency to disregard, possibly without even recognition or understanding of the security requirements has proven to be a serious challenge to software security engineering. The paradigm change, starting in the mid-1990s, coincided with the vast increase in Internet-connected programmable devices and complexity of the software. This was prominently demonstrated by Microsoft, whose Windows 95, 98 and XP operating systems suffered from an enormous amount of security flaws and vulnerabilities [6].

This lack in software security was recognized and partially remedied by new efforts to mix software engineering and security engineering at the development phase. A substantial early contribution was the introduction of Security Development Lifecycle (SDLC) by Viega and McGraw in 2002 [26]. Microsoft, learning from their failures, also published a design-centered SDLC around mid-2000s [6]. McGraw and partners later went on to introduce the Building Security In Maturity Model (BSIMM) in 2009. This model is based on industry surveys, and provides a set of best practises for software security development and organizational security processes.

2.2 Related surveys

Stavru [23] has pointed out certain shortcomings in industry surveys in the field of software engineering. The key points of their analytic framework are here applied to the BSIMM. First of all, we do not know *who answered the survey*. The BSIMM does provide industry verticals (business areas where the participant companies belong) and good demographics about them, but does not clearly state who answered and what was her role. There is a mention of 109 executives and "more than 80 individuals", but no details are provided about details. Nor is it clearly stated *what were the questions* asked. To overcome these shortcomings, our survey adheres to the guidelines and good practises presented in [23].

The background information states that the companies have had a Software Security Group (SSG) initiative in place for about four years. The BSIMM does good job in providing links of how the SSG should be constructed, what kind of experts it should contain, and how it should be led, but tells very little of individual SSGs. The median size of the group is stated to be 5 (smallest 1, largest 130), and average size is 11.9 – but on top of that, all that we know is their average age and that there are 109 of them.

Finally, BSIMM describes itself as descriptive, not prescriptive. In other words, BSIMM provides a balanced scorecard for consulting—the winners and losers of a "software security popularity contest". In BSIMM, security activities are divided into four main domains: governance, intelligence, SDLC Touchpoints, and deployment. The focus in our survey is in SDLC Touchpoints, introduced in McGraw's 2006 praiseworthy book on software security [11]; its relationship to BSIMM is further explained by McGraw in [12]. Touchpoints are SSDL practises consisting of architecture analysis, code reviews and security testing, the building blocks of software security engineering. From a software engineer's point of view this contains the development-time activities and shapes the software security engineering processes and activities. There are, however, activities involved in all four domains of BSIMM in our survey – some directly worded after the BSIMM activities, some closely related. Based on our results, however, BSIMM appears to provide a somewhat biased view on software security. This is likely due to the simplified way they have chosen to report the usage of software security activities in their survey. Also the choice of respondents is an important factor; unlike BSIMM, our survey targeted software developers in general, whereas BSIMM appears to have a strong focus on security-oriented companies, and specifically on executives.

Software development practises in general have been extensively surveyed in Finland [21] and elsewhere [10]. Security education and security skills have recently been charted in Finland [22]. On the other hand, surveys about security software development process appear almost nonexistent. Besides BSIMM's surveys and reports, there exists very little evidence on *how*—if at all—software security is actually implemented during software development. This gap is unfortunate (and somewhat odd) compared to the substantial body of empirical research on software development processes gathered over the decades of its existence. To fill this gap, and to provide comparability with the previous studies, our survey, labeled "Secure Agile Survey", was aimed to confirm the software development practises in combination of 40 suggested development-time security activities.

3 RESULTS

The results are disseminated by first discussing the survey design and the activities observed. After this brief discussion, the main empirical points are delivered by focusing on the reported use of the activities and their perceived impact on software security.

3.1 Materials

The results presented are a part of a larger web survey on security engineering practices and agile software development processes currently used in Finland. The survey is structured into three main categories: (a) the background of the respondents and their organizations; (b) the typical agile software development processes used in the organizations; and (c) the security engineering practices used. All of these categories are linked. In particular, respondents were instructed to consider only those software projects that were implemented by using both agile and security engineering processes. While keeping this point in mind, this paper focuses on a subset of the security engineering practices.

Table 1: Security activities in the survey.

Phase	Source	Activity
Requirement		
	Other	Identify user roles and requirements
	SDL	Establish security requirements
	BSIMM	Create a data classification scheme and inventory
	BSIMM	Security requirement review
	SDL	Set quality gates
	BSIMM	Translate compliance constraints to requirements
	VAHTI	Define application goal and criticality
	SDL, VAHTI	Application's security and privacy risk analysis
	VAHTI	Business impact analysis
Design		
	SDL, VAHTI, BSIMM	Threat modeling
	SDL	Design requirements established
	Other	Abuse or misuse cases
	VAHTI	Architecture and Application Development Guidelines
	VAHTI	Application security configurations specified
	VAHTI, SDL	Attack surface analysis and reduction
	VAHTI	Application Security Settings Definitions
Implementation		
	BSIMM, SDL	Use coding standards
	SDL	Approved tools
	SDL	Static analysis
	SDL	Deprecation of unsafe functions
	Other	Security specific hardening sprints
	VAHTI	External interface review
	BSIMM	Use automated tools along with manual reviews
	VAHTI	Documentation of security solutions
Verification and Validation		
	*VAHTI	Security specific test cases
	SDL	Fuzz testing
	*BSIMM	Penetration testing
	SDL	Dynamic analysis
	SDL	Attack surface review
	VAHTI	Review security testing plans
	VAHTI	Code reviews during testing
	VAHTI	Automated testing tools
Release		
	BSIMM	Code signing
	SDL	Incident response plan created
	BSIMM	Documentation required by regulations
	*VAHTI	Internal security audits
	*VAHTI	External security audits
	SDL	Formal certification
	*VAHTI	Security patch planning

Items marked with * are differently worded than corresponding BSIMM activities.

The initial questionnaire was piloted with a small set of professional security engineers and researchers. Based on their feedback, the draft questionnaire was adjusted regarding questions that were found to be hard to understand. After this iterative development, the questionnaire was posted in December 2017 via email to several Finnish software companies and engineers working in these companies. In addition, the survey was promoted online by asking people to forward it to interested parties. This promotion included also social media platforms. After a month, reminder messages were sent. The questionnaire was closed in late January 2018. In total, 62 usable responses were received.

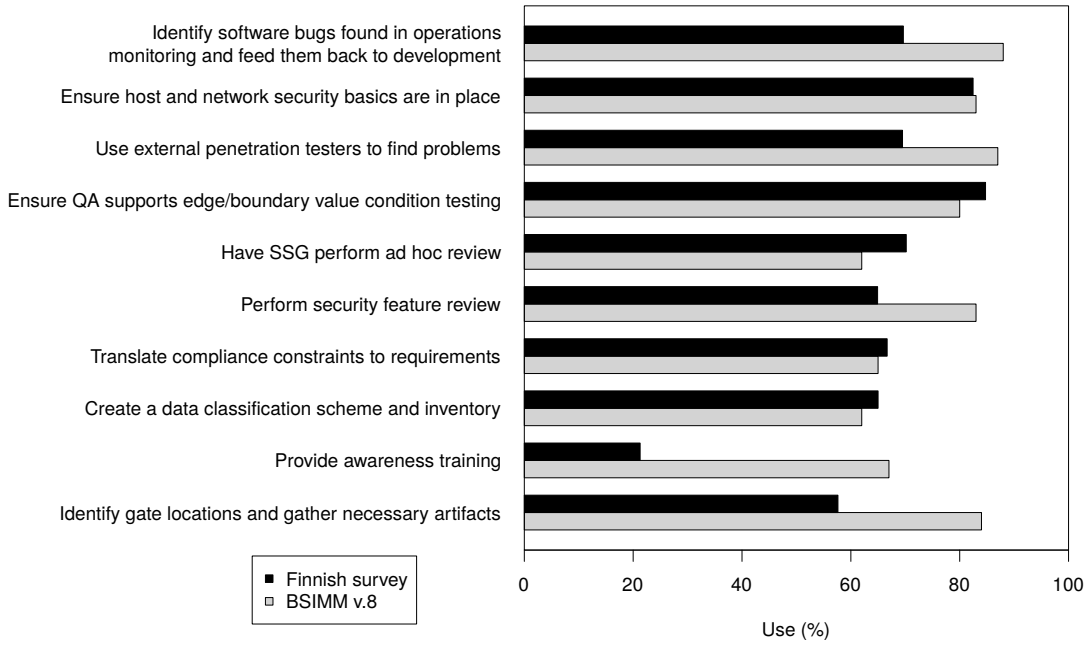

Figure 1: A Comparison of Selected Security Engineering Activities

3.2 Activities in the Survey

The surveyed security engineering activities, presented in Table 1 were drawn from established models and standards, with few additions drawn from extant literature. However, for comparing the results to BSIMM, a number of activities were explicitly included from the BSIMM version 8 [13]. The other activities surveyed were based on the Microsoft's SDL and the Finnish governmental VAHTI framework. The inclusion of VAHTI is particularly important in Finland, given the country's extensive public sector, currently undergoing a large-scale digitalization process (see e.g. [15]). VAHTI instructions for software development have been created by security professionals by collecting best practices and security activities [25].

The BSIMM promotes a "top-12" list of organizational security activities, which are expressed as "activities that 'everybody' does". According to BSIMM's argument, a security practise can be formed by selecting three activities from each of their four domains; one of these domains is a security development lifecycle model, the SDLC Touchpoints which resides directly within the scope of our survey. Microsoft does not explicitly state the criteria how the activities were selected into the SDL. the model has not received any significant changes since its creation. The focus at Microsoft appears to have been simplicity, scalability and applicability to various organizations and products [6]. VAHTI instructions for software development contain a comprehensive set of security activities, largely based on best practises; the usage and impact of VAHTI instructions for software development has been analysed in [20]. The activities in these models have a significant overlap, mainly due to the generic nature of the Microsoft SDL. BSIMM claims to be descriptive whereas VAHTI is prescriptive to the level of a *de facto* national standard.

Our survey was activity and lifecycle based and the questions were grouped by the SDLC phases. Primary purpose was measuring their usage, and for activities they had used, the respondents were asked also for an evaluation of their effectiveness. The 40 included security activities were divided into five development phases: requirement elicitation, design, implementation, verification and release. Of the 40 activities five were word-for-word picked from the BSIMM top 12 using their distinctive nomenclature. Further five were included in or similar to the activities picked from SDL or VAHTI. As we concentrated on development-time activities, user training was asked giving a yes/no option regarding the security training given by their current organization; the results may not be directly comparable.

Activities of the SDL were also prominently featured, with 11 of SDL's 15 development-time activities selected, added with the training activity with the same notes as above. Rest of the activities are selected from VAHTI, complemented with three common security activities, selected from software security engineering literature as reviewed in [19]. These activities are labeled as 'Other' in Table 1. As security activities are rather universal, there was also some overlap with certain activities stemming from multiple sources. As usage data of the other models does not exist, our comparison is necessarily specific to BSIMM.

3.3 Actual Use

Most of the ten BSIMM-related activities examined were surveyed with a Likert-scale. The only exception is a question about security awareness and training provided by the respondents' employers. This question was asked with a dichotomous scale: either security training was provided or it was not provided. A five-level scale

was used for the other activities: a given activity was used systematically, mostly, sometimes, rarely, or never. For these activities, the respondents were asked to frame their answers with respect to those software projects that used both agile processes and security engineering practices. The selected activities are the ones in the Finnish study overlapping with the BSIMM' top-12 practices, directly available for comparison.

For comparing the survey results and the usage frequencies collected from the recent BSIMM report [13], all survey answers that indicated at least some usage were collapsed into a single category. For each activity, the missing values were subsequently removed. The results are summarized in Fig. 1. The comparison should be interpreted only tentatively due to various methodological reasons, starting from the lack of details on how the BSIMM reports are assembled empirically. While keeping this remark in mind, most of the ten activities align even surprisingly well between the Finnish companies surveyed and the global software companies participating in BSIMM. However, there is a notable difference in terms of the security awareness and training programs provided in the companies—only about 16% of the survey respondents reported to have received formal security training from their employers. This observation is in stark contrast to the BSIMM results. To a lesser extent, there is a difference also in terms of identifying so-called gate locations such as milestones and other release engineering aspects. For all remaining activities, the usage can be reasonably interpreted to be similar, given the comparison's implicit but presumably wide error margin.

Another way to look at the use of the activities is to examine how frequently the activities were used. The original Likert-scales are suitable for this task. The results are summarized in Fig. 2, sans the training question with its dichotomous scale. The results are again rather similar between the ten activities with one exception. When compared to the other eight activities shown in the figure, ensuring the fundamentals of host and network security was used much more systematically. This result is not surprising. After all, doing "*software security before network security is like putting on your pants before putting on your underwear*" [13]. The Finnish respondents seem to agree with this truism.

3.4 Perceived Impact

The use of a particular security engineering activity does not mean that the activity would be particularly important for improving security. For this reason, the survey respondents were also asked to evaluate the impact of each activity upon security. As a survey cannot answer to a question about actual security of a software system, the term impact should be understood as the respondents' educated opinions on the *perceived impact* upon security. Such perceived impact was again solicited with a five-level Likert-scale, ranging from a very high impact to a very low perceived impact.

There are three noteworthy observations to make from the results summarized in Fig. 3. First and foremost: for all nine activities, the perceived impact is much higher than the corresponding use (cf. Fig. 2). In fact, all of the activities are perceived to have at least some impact upon security. Only a negligible amount of answers fall into the category of very low perceived impact. Second, the fundamental premises of host and network security match in terms of use and perceived impact. Third, penetration testing is perceived to have a very high impact, although this activity is only seldom systematically used in Finland. Excluding this interesting detail, the results largely bespeak about a mismatch between use and impact. In other words, there is still much to improve. A possible explanation for this mismatch relates to regulations and standards.

4 DISCUSSION

As a software development objective, security is typically implemented in a way that provides various types of *assurance* [24], by which it is then evaluated. Most of the software security engineering activities provide means to that end: reviews, tests, verification, and extensive documentation produce security assurance artifacts and act as evidence of security existence. The original idea behind security assurance was to prove the existence of such security mechanisms that enforce the system's security policies; in early security specifications this meant primarily programmatic evidence, i.e., machine-produced log files [4]. Over several iterations of regulation, the definition of assurance expanded to cover also various reviews and programmer-created documentation [5].

This trend is notable in both BSIMM and the Finnish survey: of the nine common activities, only two are direct security improvement activities: "Identify software bugs found in operations monitoring and feed them back to development" (BSIMM code CMVM1.2), and "Ensure host and network security basics are in place" (BSIMM code SE1.2). In the software development lifecycle, both these activities belong to phases that take place *after* development. It should be well noted that feeding the found defects from maintenance phase into the development backlog is a vital security activity in DevOps model and links the maintenance phase directly back to the design, implementation and verification phases.

The rest of the activities fall into category of security assurance: two of the activities belong into domain of security testing: Ensure QA supports edge/boundary value condition testing (BSIMM code ST1.1) and Use external penetration testers to find problems (BSIMM code PT1.1). Notably, BSIMM promotes the penetration testing activities as separate from other security testing.

The rest of the activities are various reviews. At code level, expert reviews are used to enforce coding standards and locate security design flaws and bugs; reviews are also applied to security architecture and security features. Reviews are also perceived very efficient and cost-effective ways to improve software security [24]. The main issue with reviews is that they are performed typically by external personnel, which may not be available unless explicitly required – and paid for – by the customer. This leads to two minor conclusions about the BSIMM: first, it appears regulation-driven with emphasis on security assurance and compliance requirements; second, it notably promotes use of external security experts despite one of BSIMM's basic elements is the organization's own SSG, or a "security satellite".

5 CONCLUSIONS

The foremost conclusion is clear: (a) the security engineering activities currently used in Finland align well with the BSIMM-based activities used in the global software industry. The reasons for this alignment are likely also similar. Despite increased promotion for

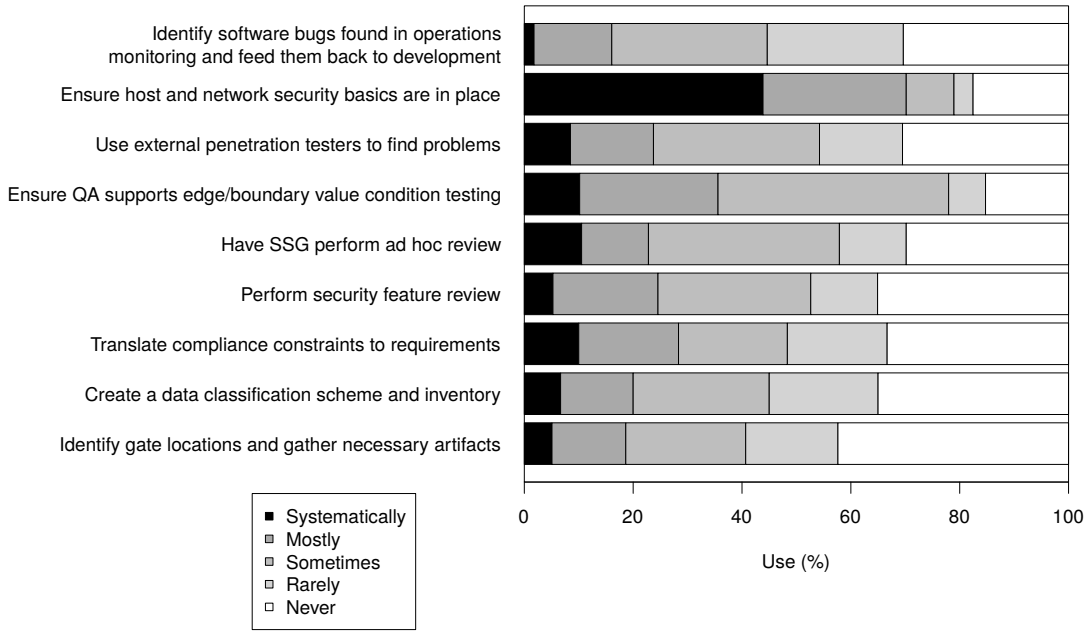

Figure 2: A Breakdown on the Use of Selected Security Engineering Activities

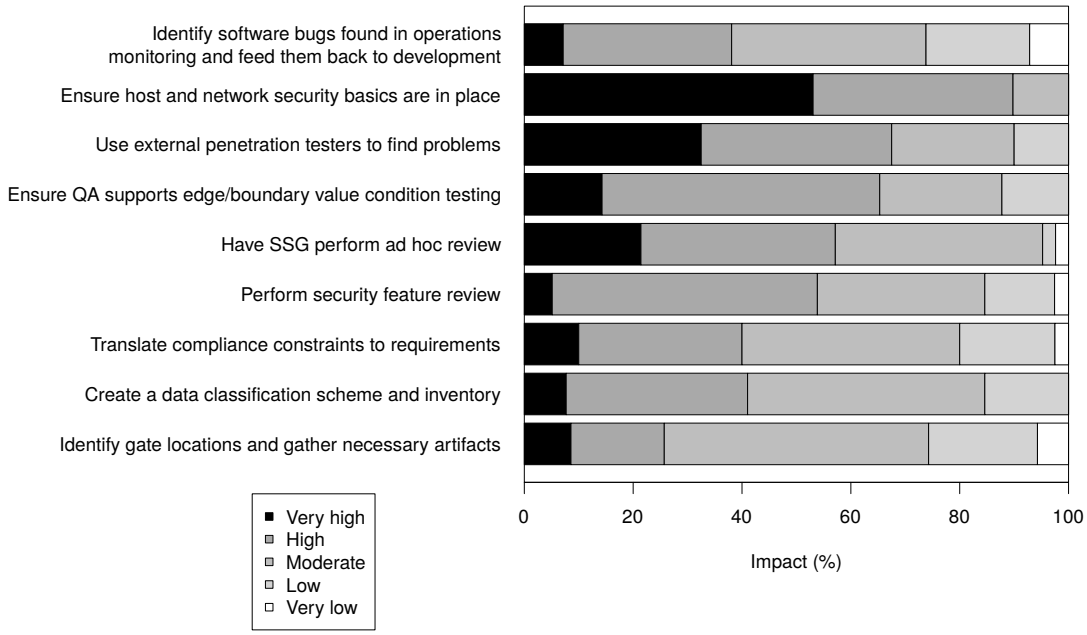

Figure 3: A Breakdown on the Perceived Impact of Selected Security Engineering Activities

security and increased recognition as an important property of good software, security engineering in software development appears still to be driven by regulation. That is, the rationale for using the majority of "top-12" activities surveyed can be plausibly explained by regulation.

The results further align rather well with a recent industry survey according to which setting standards is often more important than actually following through, external reviews are seen as important for security, and last but not least, many companies fail to perform desired security activities [16]. In particular, (b) the small set of activities surveyed are actively used in Finland, but the use is still limited compared to the perceived impact upon security. When compared to BSIMM's surveys, (c) security training is only seldom used according to the results. This observation may relate to a selection bias: according to the background information collected, most of the respondents are professionals with more than six years of software development experience. As all respondents have also explicitly worked in the domain of security engineering, formal security training may be a redundant activity for the majority of the respondents. Finally, (d) it is worth pointing out that penetration testing was perceived as particularly important for security. This result is in contrast to the mentioned survey, which rather pointed out the limitations of penetration testing when compared to code reviews [16]. Given that penetration testing is not widely used in Finland according to the results, the reason may relate to the concept of perceived impact on software security. In other words, an activity that is widely promoted by consultants and industry associations may not correlate with the actual impact of the activity. This remark leads to a couple of important points for both research and practice.

First, measuring the popularity of security activities hardly constitutes a maturity model. Instead of just providing a ranking, a maturity model should describe a tangible framework for organizations to translate their security objectives and requirements into a set measurable security activities, processes, and artifacts. Generic security models, to-do-lists, and prescriptive processes are necessary when the security objectives of the software development process, and thus the software product, are based on security regulation, laws or standards.

Second, in most software projects, security engineering activities should be arguably based on the project's threat models and risk assessments, which depend on the application area, implementation platform, and the operating environment. Instead of prescribing a SSE process, the SSDLs would be more beneficial when they contain a set of targeted security engineering activities. From these sets, the software security experts would be able to pick the most effective ones fitting their software development processes and fitting their specific security profiles.

Software security is tightly related to the privacy of the customers of an electronic business, and increasingly also all citizens in an information society. Regulations will continue to drive software security by providing a set of security objectives, but it is in the hands of software and security practitioners to define and implement the correct and most effective measures to achieve those objectives. In software industry, this means setting up processes that integrate into the development processes as seamlessly and effortlessly as possible; optimally in the way that the benefit gained from the security measures exceeds the cost. Building such a framework should be based on universal and generic building blocks and theoretical constructs. Further empirical research can suggest how to put these together in a way that is beneficial in practice.

REFERENCES

[1] Ross J. Anderson. 2008. *Security Engineering: A Guide to Building Dependable Distributed Systems* (2 ed.). Wiley Publishing.

[2] K. Beck. 1999. Embracing change with extreme programming. *Computer* 32, 10 (Oct 1999), 70–77. https://doi.org/10.1109/2.796139

[3] Edsger W. Dijkstra. 1982. *Selected Writings on Computing: A Personal Perspective.* Springer-Verlag.

[4] DoD. 1983. *TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA.* United States Department of Defence.

[5] DoD. 1994. *SOFTWARE DEVELOPMENT AND DOCUMENTATION.* United States Department of Defence.

[6] Michael Howard and Steve Lipner. 2006. *The security development lifecycle.* Vol. 8. Microsoft Press Redmond.

[7] IEEE. 1990. *IEEE Standard Glossary of Software Engineering Terminology.* 1–84 pages. https://doi.org/10.1109/IEEESTD.1990.101064

[8] ISO/IEC. 2001. *ISO/IEC 9126. Software engineering – Product quality.* ISO/IEC.

[9] ISO/IEC standard 21827. 2008. *Information Technology – Security Techniques – Systems Security Engineering – Capability Maturity Model (SSE-CMM).* ISO/IEC.

[10] Sherlock Licorish, Johannes Holvitie, Rodrigo Spinola, Sami Hyrynsalmi, Jim Buchan, Thiago Mendes, Steve MacDonnell, and Ville Leppänen. 2016. Adoption and Suitability of Software Development Methods and Practices - Results from a Multi-National Industry Practitioner Survey. In *2016 Asia-Pacific Software Engineering Conference (APSEC).* IEEE.

[11] Gary McGraw. 2006. *Software Security: Building Security In.* Addison-Wesley Professional.

[12] Gary McGraw. 2012. Software Security. *Datenschutz und Datensicherheit - DuD* 36, 9 (01 Sep 2012), 662–665. https://doi.org/10.1007/s11623-012-0222-3

[13] Gary McGraw, Sammy Migues, and Jacob West. 2017. *Building Security In Maturity Model (BSIMM), version 8.* Technical Report. BSIMM.

[14] Microsoft. 2017. Security Development Lifecycle for Agile Development. (2017).

[15] OECD. 2018. Government at a Glance 2017 – Finland Country Fact Sheet. (2018). https://www.oecd.org/gov/gov-at-a-glance-2017-finland.pdf

[16] Andy Oram. 2017. The Alarming State of Secure Coding Neglect: A Survey Reveals a Deep Divide Between Developer Aspirations for Security and Organizational Practices. (2017). O'Reilly Media, Inc. Referenced in 5th of May 2018: https://www.oreilly.com/ideas/the-alarming-state-of-secure-coding-neglect.

[17] OWASP. 2017. Software Assurance Maturity Model. (2017). https://www.owasp.org/images/6/6f/SAMM_Core_V1-5_FINAL.pdf

[18] Mary Poppendieck and Tom Poppendieck. 2003. *Lean Software Development: An Agile Toolkit: An Agile Toolkit.* Addison-Wesley.

[19] Kalle Rindell, Sami Hyrynsalmi, and Ville Leppänen. [n. d.]. Case Study of Agile Security Engineering: Building Identity Management for a Government Agency. *International Journal of Secure Software Engineering* 8 ([n. d.]), 43–57. Issue 1.

[20] Kalle Rindell, Sami Hyrynsalmi, and Ville Leppänen. 2015. Securing Scrum for VAHTI. In *Proceedings of 14th Symposium on Programming Languages and Software Tools*, Jyrki Nummenmaa, Outi Sievi-Korte, and Erkki Mäkinen (Eds.). University of Tampere, Tampere, Finland, 236–250. https://doi.org/10.13140/RG.2.1.4660.2964

[21] P. Rodríguez, J. Markkula, M. Oivo, and K. Turula. 2012. Survey on agile and lean usage in Finnish software industry. In *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement.* 139–148. https://doi.org/10.1145/2372251.2372275

[22] Reijo M. Savola. 2017. Current Level of Cybersecurity Competence and Future Development: Case Finland. In *Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings (ECSA '17).* ACM, New York, NY, USA, 121–124. https://doi.org/10.1145/3129790.3129804

[23] Stavros Stavru. 2014. A critical examination of recent industrial surveys on agile method usage. *Journal of Systems and Software* 94 (2014), 87 – 97. https://doi.org/10.1016/j.jss.2014.03.041

[24] Jose M. Such, Antonios Gouglidis, William Knowles, Gaurav Misra, and Awais Rashid. 2016. Information assurance techniques: Perceived cost effectiveness. *Computers & Security* 60 (2016), 117 – 133. https://doi.org/10.1016/j.cose.2016.03.009

[25] VAHTI 1/2013. 2013. Sovelluskehityksen tietoturvaohje. (2013). https://www.vahtiohje.fi/web/guest/vahti-1/2013-sovelluskehityksen-tietoturvaohje Referenced 8th Oct. 2017.

[26] John Viega and Gary McGraw. 2002. *Building Secure Software: How to Avoid Security Problems the Right Way* (1st ed.). Addison-Wesley.