Software Security Maturity in Public Organisations

Martin Gilje Jaatun[1], Daniela S. Cruzes[1], Karin Bernsmed[1], Inger Anne Tøndel[1], and Lillian Røstad[2]

[1] Department of Software Engineering, Safety and Security
SINTEF ICT
NO-7465 Trondheim, Norway
{martin.g.jaatun,danielac,karin.bernsmed,inger.a.tondel}@sintef.no
http://infosec.sintef.no

[2] Norwegian Agency for Public Management and eGovernment (Difi)
Oslo, Norway
lillian.rostad@difi.no
http://www.difi.no/

Abstract. Software security is about building software that will be secure even when it is attacked. This paper presents results from a survey evaluating software security practices in software development lifecycles in 20 public organisations in Norway using the practices and activities of the Building Security In Maturity Model (BSIMM). The findings suggest that public organisations in Norway excel at Compliance and Policy activities when developing their own code, but that there is a large potential for improvement with respect to Metrics, Penetration testing, and Training of developers in secure software development.

Keywords: Software Security, Secure Software Engineering, Maturity, BSIMM

1 Introduction

Society is increasingly dependent on information and communication technology (ICT). Traditionally, ICT security has primarily been about implementing security mechanisms on the system or network level. In recent times, it has become clear that it is equally important to ensure that all mechanisms of the software is secure including the code itself, i.e., develop software from scratch so that it is secure against attack [4]. This is what we call software security.

Numerous guidelines and best-practices exist, which outline processes and methodologies that can be adopted to achieve better software security. However, in practice these are only used to a limited extent and the problem of insecure software is bigger than ever [1]. We argue that organisations learn best by comparing themselves to other organisations that tackle similar challenges, rather than comparing themselves to abstract theoretical models of ideal practices for software security.

The Building Security In Maturity Model (BSIMM) [5] is a study of real-world software security initiatives that is organised so that an organisation can use it to determine where they stand with their software security initiative. BSIMM provides an overview over the security of software by mapping how it was built, what kind of activities that were carried out while it was built and by measuring a number of artefacts that were created when it was developed. BSIMM can also be used to measure how an organisation's software security efforts evolve over time.

The BSIMM study is dominated by large American companies, and the average number of developers in the studied organisations exceeds 4000. We were therefore curious to see how applicable the BSIMM activities were to smaller, non-commercial organisations in Europe, and if we could identify any discrepancies and obvious areas for improvement of the current practices in these organisations.

This paper reports on a software security maturity study in 20 Norwegian public organisations, based on the BSIMM framework. The organisations that we have studied are all part of or owned by the government or municipalities in Norway.

Both on a European and a national level there is a push towards a more efficient public sector through use of eGovernment services. eGovernment consists mainly of the digital interactions between a citizen and their government (C2G), between governments and government agencies (G2G) and between government and citizens (G2C) and between government and businesses/commerce (G2B). The move towards eGovernment, also means a move towards a more digitalized society; a society relying heavily on ICT and software-based services to function. Thus, security becomes a major concern.

The Cyber Security Strategy for Norway [6] describes current and future security challenges and points to where efforts should be focused in order to meet those challenges. The strategy, and it's accompanying action plan, suggests that a center of competence for information security in the public sector is needed. This study is part of the work done to collect information and knowledge, needed to be able to focus the work of this competence center. The study provides valuable insight, and acts as a benchmark study. The intention is to repeat the study, at intervals yet to be determined, to assess the effect of efforts to improve software security in the public sector.

The remainder of this paper is organised as follows: In Section 2 we present the theoretical background for the study, and in Section 3 we elaborate on the method employed. The results are described in Section 4, and discussed in Section 5. Section 6 concludes the paper and outlines further work. The questionnaire that was used in the study is presented in Appendix A.

2 Background

There are two well-known maturity models for software security; BSIMM [5] and OpenSAMM [7]. Both have a common origin, and have many similarities.

BSIMM and OpenSAMM are organised in a similar fashion, and contain many similar topics and activities. Both divide the software security practices into three main areas with twelve practices in total, and place the practices into three maturity levels. However, their content and fundamental idea differ. BSIMM is based on real-world data and only includes activities that are done by real companies. As such it does not aim to tell what activities should be performed, but rather what activities are actually performed by companies today. OpenSAMM is not based on real-world data directly, but rather on experience on what activities will improve software security, and that thus should be performed. But where BSIMM is based on practices of relatively large organisations that are in the forefront regarding software security, and may thus be most relevant for that type of organisations, the description of OpenSAMM clearly states that it was designed with flexibility in mind, and should thus be useful for any type of organisation, big or small.

We chose to base our study on BSIMM rather than OpenSAMM for two reasons: we were more familiar with BSIMM; and BSIMM is a more descriptive methodology that basically is designed to measure, while OpenSAMM has a stronger prescriptive focus, i.e., to define “the right way to do it”.

2.1 OpenSAMM

The Software Assurance Maturity Model (SAMM or OpenSAMM) is an open software security framework divided into four business functions: Governance, Construction, Verification and Deployment. Each business function is composed of three security practices, as shown below:

Governance: Strategy & Metrics; Policy & Compliance; Education & Guidance

Construction: Threat Assessment; Security Requirements; Secure Architecture

Verification: Design Review; Code Review; Security Testing

Deployment: Vulnerability Management; Environment Hardening; Operational Enablement

Each practice is assessed at a maturity level from 1 to 3 (plus 0 for “no maturity”), and for each maturity level there is an objective and two activities that have to be fulfilled to achieve that level.

2.2 BSIMM

The Building Security In Maturity Model (BSIMM) measures which software security activities are included in an organisation’s overall Secure Software Development Lifecycle (SSDL). A central concept in BSIMM is the Software Security Group (SSG), which is the person (or persons) responsible for software security in an organisation. The SSG can be as small as a single person, it need not be a formal role, and need not be a full-time position. In addition, there is the concept of "the satellite"; a more or less well-defined group of developers who are not part of the SSG, but still have a special interest in and knowledge of software security, and thus can operate as the extended arm of the SSG in many contexts.

The purpose of BSIMM is to quantify the software security activities performed in real software development projects in real organisations. As these projects and organisations use different methodologies and different terminology, it is necessary to use a framework that allows describing all initiatives in a unified manner. The BSIMM framework consists of twelve practices organised into four domains; Governance, Intelligence, SSDL Touchpoints and Deployment (see Table 1). Each practice has a number of activities on three levels, with level 1 being the lowest maturity and level 3 is the highest. For example, for practice Strategy and Metrics, SM1.4 is an activity on level 1, SM 2.5 is an activity on level 2, and SM 3.2 is an activity on level 3.

Table 1. The BSIMM Software Security Framework

Governance	**Intelligence**	**SSDL Touchpoints**	**Deployment**
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

The starting point for the first BSIMM survey in 2008 [5] was to study the software security activities performed by nine selected companies. The nine companies were presumably far ahead in software security, and the activities that were observed here formed the basis of the framework in Table 1. Representatives from Cigital [3] physically visited each company, and these first surveys were done by Gary McGraw and Sammy Migues personally, using a whole day for each company.

3 Method

In this work we have performed a survey using a questionnaire with individual follow-up interviews [8]. The questionnaire (see Appendix A) is based on the BSIMM software security framework as documented in the BSIMM V report [5]. The main function of BSIMM is to serve as a yardstick to determine where an

[3] `http://www.cigital.com`

organisation stands compared with other organisations [5]. The questionnaire tells us what activities the organisation has in place, and based on how well they cover the various practices, we can determine the maturity level of each organisation. BSIMM contends that a radar chart according to the high watermark method (based on three levels per practice) is sufficient to give a rough, overall picture of maturity. We have chosen to also develop two complementary maturity measures that can provide a more balanced view of maturity. The three maturity measures we use are thus:

Conservative maturity (Scale 0-3): Here an organisation is approved at a maturity level only if all the activities in the level are met ("Yes"), provided all the activities on the lower level are also fulfilled. If the organisation performs some (but not all) activities at a level this is indicated with a "+", i.e., if you have 3 of 5 activities on the first level, the result is 0+; if all the activities at level 1 are fulfilled, and 2 out of 4 activities at level 2 are fulfilled, the result is 1+, etc. In connection with calculating the average value, a "+" is counted as 0.5. As will be seen in Section 4, since few organisations in our study do all the activities at level 1, many end up in the category 0+.

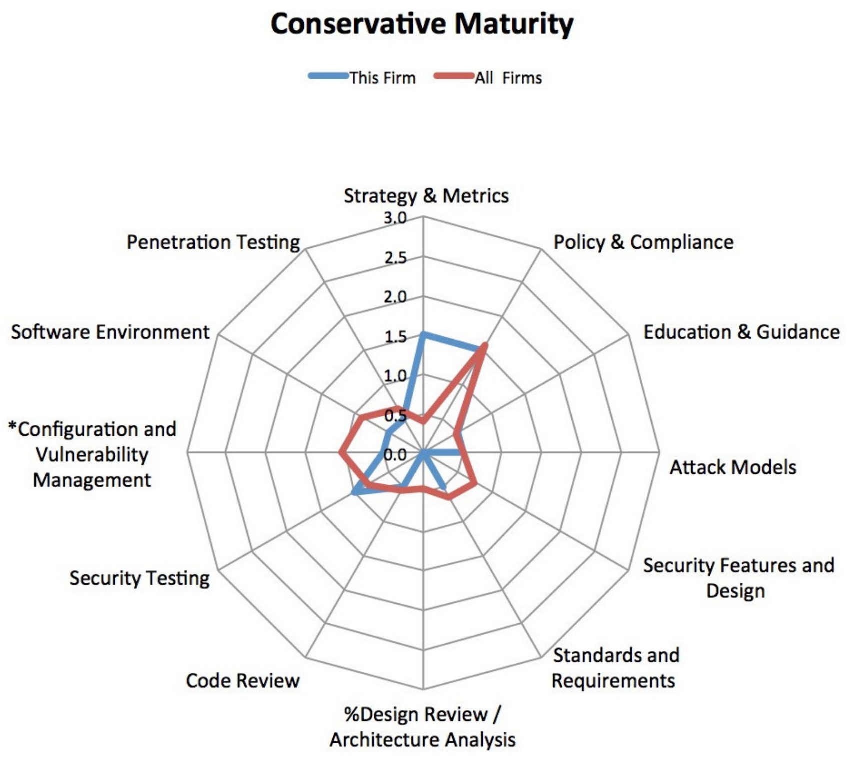
Weighted maturity (Scale 0-6): This value gives a greater emphasis for activities at a high level, even if the lower level activities are not fully implemented. The value is calculated using the following formula:

$$\sum_{i=1}^{3} \frac{\text{Observed activities at level } i}{\text{Total number of activities at level } i} \times i$$

High Watermark Maturity (Scale 0-3): This value is calculated in the same manner as in BSIMM [5]; if the organisation has at least one activity at level 3, it gets the maturity level 3. The high watermark maturity level therefore only says something about what is the level of the highest rated activity they perform. In contrast to the conservative maturity level, it is therefore easier to reach a level 2 or 3 high watermark maturity level.

In our study it will be of most interest to compare the two first maturity measures (conservative and weighted) from a given organisation with the average values of all the studied organisations to see how they compare, as we have done in the radar diagrams in Fig. 1.

We distributed the questionnaire in Appendix A in January 2015 via email to 32 Norwegian public organisations which we had reason to believe had ongoing software development activities. 20 of these organisations returned fully filled-out questionnaires. For seven of the responses, the questionnaire had been filled out in cooperation by representatives involved in software development and in general IT security work. In the other cases, the response was made either by people working on information security or on IT in general (six responses), by people working on software development (five responses), or the main responsibility of the respondent was unclear based on the job title (two responses). In most cases, at least one of the respondents had a managing role in the organisation,



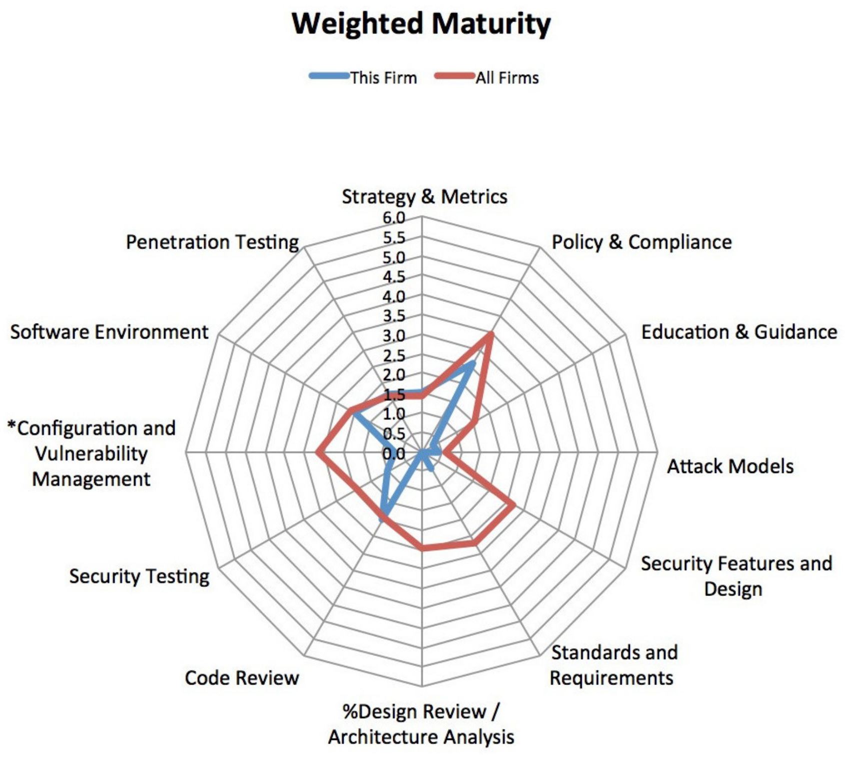


Fig. 1. Comparing an imaginary organisation with average of all organisations

e.g., information security manager, IT manager, group leader or architect. In order to verify the answers and to clarify possible misunderstandings, we organised follow-up interviews with all the involved organisations during which their answers were scrutinised and corrected whenever needed. The results were then compiled and the conservative, weighted and high watermark maturity measures were computed and analysed.

4 Results

In this section we present a selected set of the results from the study. For the full results, the reader is referred to the report [3].

The organisations with the lowest maturity level declared that it performed 9 activities of 112, while the organisation with the highest level of maturity performed 87 activities. Based on the boxplot chart in Fig. 2, we see that most of the organisations come halfway up the scale; they perform on average 39% of the activities.

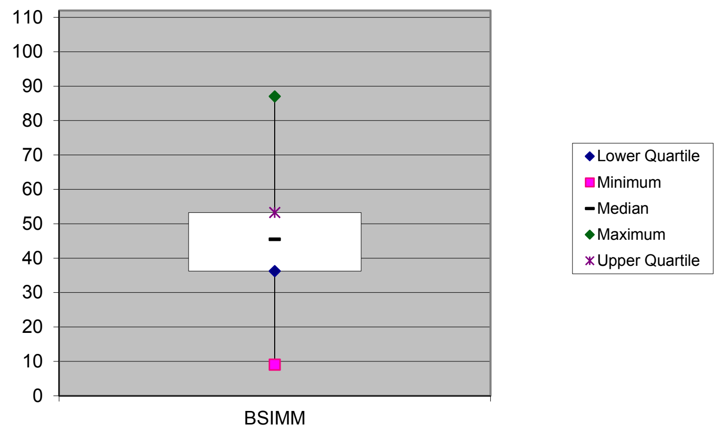


Fig. 2. The distribution of the total number of activities for all the organisations

The total number of activities for each of the organisations is shown in Fig. 3. This figure shows the "raw score" for each organisation that participated in the study.

4.1 Practices with a high degree of maturity

As can be seen in Fig. 4 and Table **??**highlevel-table, we found the highest degree of maturity among the surveyed organisations ("Observed Difi") within the practice Compliance and Policy ("Guidelines and compliance with laws and

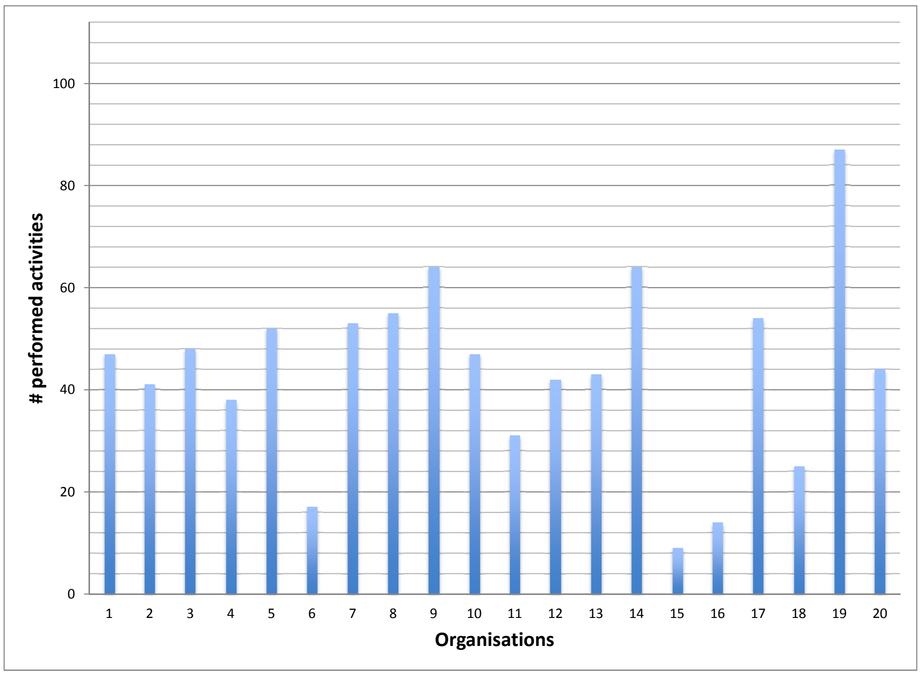


Fig. 3. The total number of activities ("raw score") for each of the organisations

Table 2. High-level results

Areas	**Observed Difi**	**Observed BSIMM**
Strategy and Metrics	33%	45%
Compliance and Policy	66%	43%
Training	27%	24%
Attack Models	25%	29%
Security Features and Design	45%	41%
Standards and Requirements	48%	40%
Architecture Analysis	39%	35%
Code Review	35%	28%
Security Testing	32%	30%
Configuration Management and Vulnerability Management	46%	43%
Software Environment	41%	42%
Penetration Testing	36%	49%

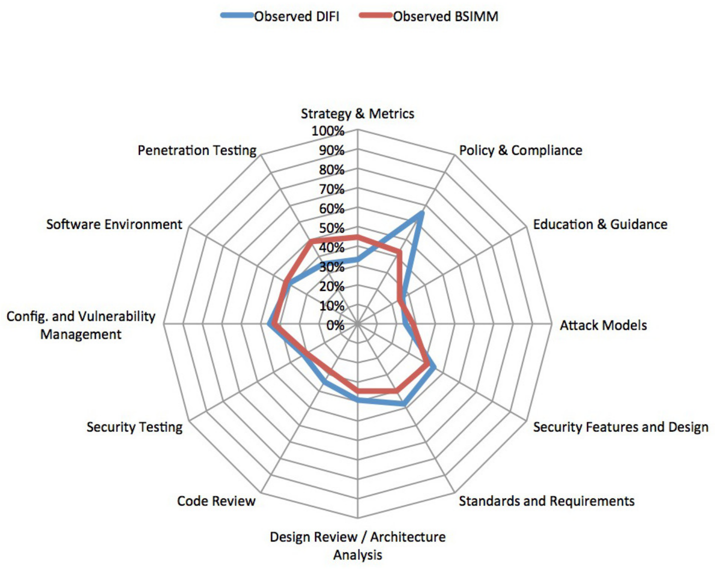


Fig. 4. High-level results illustrated

regulations"); more than 80% of the respondents answered yes to most of the activities in this area. The result is not surprising, since it concerns public organisations in Norway, which usually are accustomed to adhere to standards and government requirements. It is apparent that there is better adherence to these practices among Norwegian public organisations than the average from the official BSIMM study [5] ("Observed BSIMM"). It is also important to see the positive maturity values in this area in the context of how businesses are organised. Many public organisations have their own lawyers who handle compliance, and have a good overview of the requirements of laws and regulations. However, it is not thus given that this expertise is applicable in software development projects. In many cases there might be quite a long distance (physically and organisationally) from the internal expertise related to laws and regulations to the developers or the hired consultants that are central to software development.

Three other practices that received high maturity are Construction and intelligence, Security features and design, and Standards and Requirements). 80% of the organisations say they do SFD1.2 ("Security is a regular part of our organisation's software architecture discussion") and 80% say that they do SR 2.3 ("We use a limited number of standard technology stacks"). Regarding the latter, in most cases this meant that the organisation uses only Microsoft products (Microsoft Active Directory, Microsoft Internet Information Services, etc.) in their software development and production processes.

In the practice Configuration Management and Vulnerability Management, 85% of the organisations said that they satisfy CMVM1.1 ("The software security group has procedures for incident response, in collaboration with the incident response team (if it exists)"). They also claim that they do CMVM 2.2 ("We track software DEFECTS found during operations until they are closed"), but it seems as though many people equate this with their internal bug tracking system that often does not take particular account of security flaws. In such cases it is necessary that the security flaws are prioritized high enough that they *must* be handled; if not, there is no guarantee that they actually closed within a reasonable time. It is also unclear to what extent there are procedures for cooperation during security incidents, but respondents says that developers will be able to get involved when needed.

Finally, 90% of the organisations said that they meet SE1.2 ("We use accepted good practice mechanisms for host/network security"), but this is not so surprising since this is strictly not about software security. Security seems to have a relatively large focus amongst the people involved with the network, operations and infrastructure. There are indications, however, that there is a distinction between the developers and the "security people" in the organisations. One respondent stated that security can be perceived as an obstacle among their developers, since those who work with security might restrict too much traffic through firewalls etc. There is a different culture among those who work with infrastructure than among those who are involved in the software development.

4.2 Practices with a low degree of maturity

According to our results, the area with the lowest maturity is Attack models, followed closely by Strategy and Metrics. Regarding attack models, 80% stated that they do AM1.5 ("The software security group keeps up to date by learning about new types of attacks / vulnerabilities"), and 55% said they do AM 1.6 ("Build an internal forum to discuss attacks") but all the other activities in the AM practice are performed by fewer than 25% of the organisations.

One reason that these activities are performed only to a small extent may be that these are activities that are very specifically related to security and therefore come in addition to, or on top of, all the other activities that are being done in the development process. During the follow-up interviews, several of the organisations said they are fully aware that these are areas where they have a potential for improvement. Regarding the two activities mentioned above that relatively many do, several of the respondents indicated that these are largely done outside the development environments. Those who work with operations and infrastructure often get alerts or information on new attacks, and these are discussed as needed. Respondents assume then that developers will be notified of things that are relevant to them, but there seems to be little systematic effort on monitoring attacks in the development environment. Several respondents said that individual developers are adept at keeping up to date also in the security field, for example, to gain knowledge about issues related to components they use themselves, but this work seems to be relatively unstructured and largely depends on the individual developer.

When it comes to work with strategy, there are some organisations that have started some activities related to this, but few organisations currently have a strategic and systematic approach to software security, where they clearly assign responsibility, make plans and strategies, and follow up the implementation and effectiveness. This could relate to the fact that few companies have a clear answer to the question of who are the SSG. All the organisations do some activities related to software security, and some do many, but since this is not done systematically, it is difficult to say something about the effect of the work done in organisations today related to this. On the basis of this investigation, we can not say anything about the reasons for this, but some statements in the interviews indicate that there is little awareness of the importance of software security within the management. As an example, in one interview it was stated that risk at the enterprise level was not relevant to software development.

4.3 Result Summary

To summarize, there is considerable variation in the maturity of the various organisations in the study. As can be seen in Fig. 2, the most mature organisation has implemented 87 of the 112 activities, and the least mature has only implemented 9. The average of the 20 businesses is about 44 of 112. If we look at the three most mature businesses as illustrated in Fig. 5, we notice that, even though the conservative maturity level for the practice Strategy and Metrics is consistently low amongst all the top three organisations, there is an extreme variation in the practice Code Review (ranging from 0.5 to 3).

5 Discussion

As became apparent during the follow-up interviews, the organisations that participated in the study vary as to how much development they perform themselves, and to what extent they use external consultants. A few rely solely on external consultants, while many operate with a mix of internal and external developers. Some largely purchase solutions from system vendors, and make adjustments internally, while others develop the bulk of the solutions themselves. This affects how they work with security, and also the extent to which they have an overview of and control over their software security activities. Some of the respondents had acquired input from their vendors on the questionnaire. Others had little overview of what the vendors did in terms of activities related to information security. This uncertainty applies both to training related to contractor developers as well as to what activities are performed in the development process. The answers to the survey did not indicate that there are any differences in the maturity levels that are due to how the software development teams are organised. However, it is clear that some of the organisations who participated in the study have relatively high levels of expertise, resources and experience related

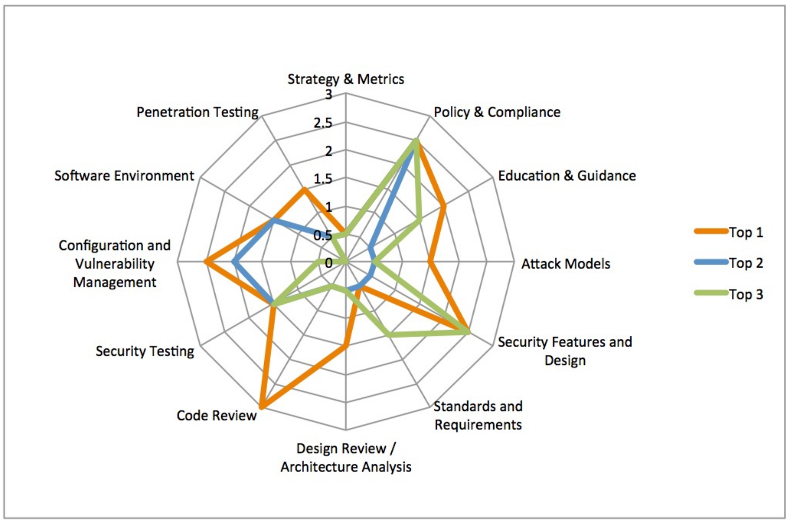


Fig. 5. Conservative and weighted maturity for the three most mature organisations

to software development in general, while others have less experience with this. At the same time, our results also show that there are many good practices also among those who do not have a large number of developers. The two organisations that received the highest maturity scores (weighted maturity) have 10 to 20 developers in total (internal and contracted).

The BSIMM framework is based on the idea that there is a formally defined software security group (SSG), and the activities are centered around this group. Few of the surveyed organisations had such a formally defined group. Several organisations have a manager with more or less explicit responsibility for software security, but then usually as part of an overall security responsibility in the organisation.

The method used in our study can be characterized as “assisted self-evaluation”; the respondents from the various organisations indicated in a questionnaire which software security activities that they do, and then they participated in a follow-up interview with the purpose of clarifying uncertainties and correcting possible errors in the questionnaire. During the interviews, many of the respondents mentioned that they found it difficult to answer some of the questions in the questionnaire. In some cases this was because they did not understand what certain activities entailed, for example because they were not familiar with the concepts and terminology that were used. In other cases, they lacked knowledge about the practices in their own organisation or among their consultants and vendors. In several cases, however, the uncertainty was linked to the challenge of responding a simple “yes” or “no” to whether they perform a certain activity. These cases were discussed in depth in the follow-up interviews, aiming to reach a most equal assessment of the various organisations. In most cases the follow-up interviews resulted in some changes to the original answers in the questionnaire. It is also important to point out that some of the respondents seemed to have different attitudes to the study. Some appeared keen to put forward as much as possible of what they do in order to get a good score, while others were more modest on their own behalf; feeling uncomfortable about the possibility that they might claim to do an activity that they did not implement to the full extent.

Another uncertainty factor in our study is that the follow-up interviews were conducted by three different researchers. However, they did synchronize their assessment criteria, both before and during the interview phase, in order to ensure that they had an as similar as possible perception of what is required to receive a "yes" for the various activities in the questionnaire. However, it is still possible that researchers may have made different assessments related to what should be approved as an activity.

Since the study is based largely on self-evaluation, there is reason to believe that the resulting "BSIMM-score" in our study is higher than it would be with a review in line with the one made by Cigital in the original BSIMM study [2], since we have not been in a position to verify the claims made by each organisation. In concrete terms, this implies that we must assume that it has been easier for the organisations to get an activity "approved" in our study than it would be if Cigital had done the survey in accordance with its usual practice. This means that although our results provide some indications of the maturity level of the evaluated organisations, none of the organisations in our study can claim that they have established their "BSIMM Score". It would also be misleading to compare their results directly with the official BSIMM reports. On the other hand, the validity of the answers in our study were increased because of the follow-up interviews, compared with the results from a pure survey. Furthermore, using a questionnaire approach significantly lowers the threshold for initiating a software security maturity study, and we maintain that it is a useful exercise for determining a baseline.

BSIMM claims to be descriptive rather than normative, but by ranking activities in maturity levels, there is an implicit statement that some activities are "better" (or more mature) than others. However, a given organisation may have good reasons for not doing a certain activity, but this will not be reflected in the results from our study. Sometimes checklists have an option to specify "Not relevant" to a given question, and it could be worth considering adding this to the BSIMM yardstick as well.

6 Conclusion and Further Work

This study shows that the public organisations that we have studied are doing a number of activities that contribute to security in the software they are developing. However, it is clear that few are working strategically with software security, where they have a comprehensive and systematic approach and follow up using metrics to evaluate the effectiveness of the various activities. Many of the organisations are very dependent on the interest, competence and initiative of individual developers when it comes to keeping up to date on software security and ensuring that security is not forgotten in the development lifecycle. This stands in contrast to the operations or network side of organisations, where security seems to have a clear priority.

Most of the organisations that participated in our study seemed to be very interested in the topic of software security, but many point out that they have limited resources and this is also reflected in the results. Some stated clearly that they have prioritized other areas in their effort to improve security, however, they all seem to realise the importance of addressing software security as well.

In order to put public organisations in a position to work strategically and systematically with software security, it is important to implement training activities on this topic. Thus far, software security seems to be a very small part of efforts to increase knowledge and awareness of information security in the various organisations.

Several respondents commented that they found many of the governance-related activities more difficult to perform when using an agile development method. This would indicate that there is a need for further study on how to ensure software security in an agile environment. It would also be interesting to compare our results with a similar study on private sector organisations in Norway, which is something we hope to be able to initiate later this year.

Acknowledgment

The research reported in this paper was commissioned by the Norwegian Agency for Public Management and eGovernment (Difi), and partially funded by the EU FP7 project OPTET, grant number 317631.

References

1. Hope, P.: Why measurement is key to driving improvement in software security. Developer Tech (2014), http://www.developer-tech.com/news/2014/mar/06/why-measurement-key-driving-improvement-software-security/
2. Jaatun, M.G.: Hunting for Aardvarks: Can Software Security Be Measured? In: Quirchmayr, G., Basl, J., You, I., Xu, L., Weippl, E. (eds.) Multidisciplinary Research and Practice for Information Systems, Lecture Notes in Computer Science, vol. 7465, pp. 85–92. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-32498-7_7
3. Jaatun, M.G., Tøndel, I.A., Cruzes, D.S.: Modenhetskartlegging av programvaresikkerhet i offentlige virksomheter. Tech. Rep. A26860, SINTEF ICT (2015), http://sintef.no/publikasjoner/publikasjon/?pubid=SINTEF+A26860
4. McGraw, G.: Software security. Security & Privacy, IEEE 2(2), 80–83 (Mar 2004)
5. McGraw, G., Migues, S., West, J.: Building Security In Maturity Model (BSIMM V) (2013), http://bsimm.com
6. Norwegian Ministries: Cyber Security Strategy for Norway (2012), https://www.regjeringen.no/globalassets/upload/fad/vedlegg/ikt-politikk/cyber_security_strategy_norway.pdf

7. OpenSAMM: Software Assurance Maturity Model (SAMM): A guide to building security into software development, http://www.opensamm.org/
8. Robson, C.: Real World Research. John Wiley & Sons, 3 edn. (2011)

A Questionnaire

The questionnaire is taken from the BSIMM activity descriptions [5], with only minor textual modifications. Note that the official BSIMM study does not rely on questionnaires. As mentioned before, BSIMM ranks activities on three levels, but we decided not to show this to the respondents; we also re-arranged some activities, placing variations on the same activity together even though they are on different levels in the BSIMM description.

A.1 Governance

Strategy & Metrics

- We publish our process for addressing software security; containing goals, roles, responsibilities and activities.
- We have a secure software evangelist role to promote software security internally.
- We educate our executives about the consequences of inadequate software security.
- We have identified gate locations in our secure software development process where we make go/no go decisions with respect to software security.
- We enforce the identified gate locations in our secure software development process where we make go/no go decisions with respect to software security, and track exceptions.
- We have a process of accepting security risk and documenting accountability. In this process we assign a responsible manager for signing off on the state of all software prior to release.
- The software security group publishes data internally on the state of software security within the organisation.
- In addition to the software security group, we have also identified members of the development teams that have a special interest in software security, and have a process for involving them in the software security work.
- We have identified metrics that measure software security initiative progress and success.
- The software security group has a centralized tracking application to chart the progress of all software.
- The software security group advertises the software security initiative outside the organization (for example by writing articles, holding talks in conferences, etc).

Policy & Compliance

- The software security group has an overview of the regulations that our software has to comply with.
- We have a software security policy to meet regulatory needs and customer demands.
- The software security group is responsible for identifying all legislation related to personally identifiable information (for example personopplysningsloven).
- We have identified all the personally identifiable information stored by each of our systems and data repositories.
- All identified risks have to be mitigated or accepted by a responsible manager.
- We can demonstrate compliance with regulations that we have to comply with.
- We make sure that all vendor contracts are compatible with our software security policy.
- We promote executive awareness of compliance and privacy obligations.
- We have all the documentation necessary for demonstrating the organisation's compliance with regulations we have to comply with (for ex. written policy, lists of controls, artifacts from software development).
- When managing our third party vendors, we impose our software security policies on them.
- Information from the secure software development process is routinely fed back into the policy creation process.

Education & Guidance

- We have a security awareness training program.
- We offer role-specific security courses (for example on specific tools, technology stacks, bug parade).
- The security awareness training content/material is tailored to our history of security incidents.
- We deliver on-demand individual security training.
- We encourage security learning outside of the software security group by offering specific training and events.
- We provide security training for new employees to enhance the security culture.
- We use the security training to identify individuals that have a particular interest in security.
- We have a reward system for encouraging learning about security.
- We provide security training for vendors and/or outsourced workers.
- We host external software security events.
- We require an annual software security refresher course.
- The software security group has defined office hours for helping the rest of the organization.

A.2 Construction / Intelligence

Attack Models

- We build and maintain a top N possible attacks list.
- We have a data classification scheme and an inventory of attacks so we can prioritize applications by the data handled by them.
- We maintain a list of likely attacker profiles.
- We collect and publish attack stories.
- The software security group keeps up to date by learning about new types of attacks / vulnerabilities.
- We have an internal forum to discuss attacks.
- We link abuse cases to each attacker profile.
- We have a list of technology-specific abuse cases.
- We have an engineering team that develops new attack methods.
- We have automated the attack methods developed by our engineers.

Security Features and Design

- Our software security group builds and publishes a library of security features.
- Security is a regular part of our organization's software architecture discussion.
- The software security group facilitates the use of secure-by-design middleware frameworks/common libraries.
- The software security group is directly involved in the design of security solutions.
- We have a review board to approve and maintain secure design patterns.
- We require the use of approved security features and frameworks.
- We find and publish mature design patterns from the organization.

Standards and Requirements

- The software security group create standards that explain the accepted way to carry out specific security centric operations.
- We have a portal where all security related documents are easily accessible.
- The software security group assists the software development team in translating compliance constraints (for instance from legislation) into application specific security requirements.
- We use secure coding standards in our software development.
- We have a standards review board to formalize the process used to develop security standards.
- We use a limited number of standard technology stacks.
- We have a template SLA text for use in contracts with vendors and providers, to help prevent compliance and privacy problems.
- We have procedures to communicate and promote our security standards to vendors.
- We have a list of all open source components used in our software.
- We manage the risks related to using open source components.

A.3 Verification / Touchpoints

Design Review / Architecture Analysis

- We perform security feature review.
- We perform design review for high-risk applications.
- We have a software security group that leads review efforts.
- We use a risk questionnaire to rank applications in terms of the risk they are exposed to.
- We have a defined process to do architecture analysis.
- We have a standardized format for describribng architecture that also covers data flow.
- The software security group is available to support architecture analysis when needed.
- The software architects lead design review efforts to detect and correct security flaws.
- Failures identified during architecture analysis are used to update the standard architecture patterns.

Code Review

- We create a list with top N software security defects list.
- The software security group does ad-hoc code reviews.
- We use automated tools (such as static analysis) along with manual review to detect software security defects.
- We make code review mandatory for all projects before release.
- The software security defects found during code review are tracked in a centralized repository.
- We enforce coding standards to improve software security.
- We have mentors for code review tools for making most efficient use of the tools.
- We use automated tools with tailored rules to improve efficiency and reduce false positives.
- We combine assessment results so that multiple analysis techniques feed into one reporting and remediation process.
- When a software defect is found we have tools to search for that defect also in the whole codebase.
- We perform automated code review on all code to detect malicious code.

Security Testing

- We perform adversarial tests with edge and boundary values.
- We create our tests based on existing security requirements and security features.
- We integrate black box security tools into the testing process (including protocol fuzzing).
- We share security test results with QA.
- We include security tests in QA automation.
- We perform fuzz testing customized to application APIs.
- We base the security tests on the security risks analysis.
- We use code coverage tools to ensure that security tests cover all parts of the code.
- We write tests cases based on abuse cases provided by the software security group.

A.4 Deployment

Configuration and Vulnerability Management

- The software security group has procedures for incident response, in collaboration with the incident response team (if it exists).
- We are able to make quick changes in the software when under attack.
- We perform drills to ensure that incident response capabilities minimize the impact of an attack.
- We identify software defects found in operations (for ex. by intrusion detection systems) and feed back to development.
- We track software defects found during operations until they are closed.
- We maintain a matrix of all installed applications in order to identify all places that need to be updated when a piece of code needs to be changed.
- When a software defect is found in a piece of code during operations we have a process to search for that defect also in the whole codebase.
- We do software security process improvement based on the analysis of cause of software defects found in operations.
- We have a system for paying rewards to individuals who report security flaws in our software.

Software Environment

- We monitor the input to software we run in order to spot attacks on our software.
- We use accepted good practice mechanisms for host/network security.
- The software security group creates and publishes installation guides to ensure that our software is configured securely.
- We create digital signatures for all binaries that we deliver.
- We use code protection such as obfuscation to make reverse engineering harder.
- We monitor the behavior of our software looking for misbehavior and signs of attacks.

Penetration Testing

- We use external penetration testers on our software.
- Defects found in penetration testing are inserted in our bug tracking system and flagged as security defects.
- We use penetration testing tools internally.
- The penetration testers have access to all available information about our software (for example: the source code, design documents , architecture analysis results and code review results).
- We periodically perform penetration tests on all our software.
- We use external penetration testers to do deep-dive analysis for critical projects to complement internal competence.
- The software security group has created customized penetration testing tools and scripts for our organization.