HAVOSS: A Maturity Model for Handling Vulnerabilities in Third Party OSS Components

Nikbakht Bideh, Pegah; Höst, Martin; Hell, Martin

Published in:
Product-Focused Software Process Improvement

DOI:
10.1007/978-3-030-03673-7_6

2018

HAVOSS: A Maturity Model for Handling Vulnerabilities in Third Party OSS Components

Pegah Nikbakht Bideh[1], Martin Höst[2], and Martin Hell[1]

[1] Lund University, Department of Electrical and Information Technology, Sweden
{pegah.nikbakht_bideh, martin.hell}@eit.lth.se
[2] Lund University, Department of Computer Science, Sweden
martin.host@cs.lth.se

Abstract. Security has been recognized as a leading barrier for IoT adoption. The growing number of connected devices and reported software vulnerabilities increases the importance firmware updates. Maturity models for software security do include parts of this, but are lacking in several aspects. This paper presents and evaluates a maturity model (HAVOSS) for handling vulnerabilities in third party OSS and COTS components. The maturity model was designed by first reviewing industry interviews, current best practice guidelines and other maturity models. After that, the practices were refined through industry interviews, resulting in six capability areas covering in total 21 practices. These were then evaluated based on their importance according to industry experts. It is shown that the practices are seen as highly important, indicating that the model can be seen as a valuable tool when assessing strengths and weaknesses in an organization's ability to handle firmware updates.

Keywords: Maturity model · Software security · Software maintenance · Firmware updates · Vulnerabilities

1 Introduction

Software maintenance focuses on issues such as adapting systems to changed functionality requirements, changed environments and corrections of faults. Identified security vulnerabilities can be seen as one type of fault that has received more attention in the last decade, following a number of attacks impacting large organizations, customers, and the society. Attacks have also become more sophisticated and the increasing number of software intensive systems, in IT systems and in the shift towards cloud computing and IoT devices, have resulted in more attack targets. The number of security vulnerabilities reported and recorded in the NVD CVE (National Vulnerability Database, Common Vulnerabilities and Exposures) database has for many years been relatively stable, ranging between approximately 4,200 to 7,900 between 2005-2016 [15]. In 2017 the number increased to about 14,700. New companies producing connected devices, as well as older mature companies adjusting their products to the current trend of connectivity, increases the competition on the market. Meeting this competition requires high functionality and fast time-to-market. Using open source software (OSS) instead of developing all code in-house decreases development time, and software maintenance for OSS can focus more on updating the software when new versions are released. However, urgent updates as a result of security vulnerabilities can be very costly for the organization. Thus, it is important to handle the process of identifying and evaluating new vulnerabilities, and subsequently update the software, in an accurate and efficient way.

A maturity model can be seen as a tool that helps organizations improve the way they work, typically by introducing and implementing changes in the organization. This is often a slow process, requiring efforts and resources throughout the organization. The change, to be effective, must have support both from management and the work force, and internal communication processes must be well implemented in order to support the change management required for implementing improvements. The maturity model will help the organization identify the issues in need for improving and prioritizing the efforts. It will also help the organization in ensuring that no important aspects are neglected. However, it will typically not in detail describe *how* the changes should be implemented since this can vary widely between organizations and depend on size, type of organization, business domain, regulations etc.

The goal of this paper is to define, evaluate, and present a maturity model (HAVOSS – HAndling Vulnerabilities in third party OSS) focusing on managing vulnerabilities in third party libraries and code, and the subsequent software update activities that are required to limit a product's exposure to attacks. We target all practices related to this aspect of software maintenance for embedded systems. The model builds upon existing models for software maintenance and security, interviews with industry, and recently published guidelines and recommendations for security in IoT devices. An initial version has been iterated using feedback from industry representatives, and has then been evaluated by industry. The evaluation shows that the proposed practices are highly relevant.

The paper is organized as follows. In Section 2 we first present related work and maturity models focusing of secure software. In Section 3, we describe the methodology used when defining and evaluating the model, and in Section 5 the different maturity levels are defined. Then we present the results of our evaluation in Section 6 and the paper is concluded in Section 7.

2 Related Work

The Capability Maturity Model for Software, CMM, and CMMI (e.g. [18]) have been very influential in how to support process improvement in software engineering. The models guide an organization through five maturity levels where process standardization (level 3) is seen as more mature than project level processes (level 2), and experience based improvement (level 4 and level 5) is a natural continuation after that type of standardization. The idea of standardizing approaches in the organization, and after that to improve through experiences, has influenced the model presented in this paper. There is also the

Software maintenance maturity model (SMmm) [1] addressing unique activities of software maintenance, and there are maturity models for process improvement implementation [14].

There are several well-known maturity models focusing on software security and the software development life cycle.

The Building Security in Maturity Model (BSIMM) [10] is based on actual practices in a large number of companies. It thus represents the current state of software security. It can be used to assess the Secure Software Development Lifecycle (SSDL). BSIMM covers 12 practices divided into the four main domains *Governance*, *Intelligence*, *SSDL Touchpoints*, and *Deployment*.

OWASP Software Assurance Maturity Model (SAMM) [2] is an open framework developed by OWASP, with the aim to help organizations evaluating their existing software security practices throughout the whole organization. SAMM is a flexible model that is designed to be utilized by both small, medium, and large companies. SAMM is built on business functions of the software development life cycle, and each business function is tied to three security practices. The business functions are *Governance*, *Construction*, *Verification*, and *Operations*.

The Systems Security Engineering – Capability Maturity Model (SSE-CMM) [9] is intended to be used as a tool to evaluate and assess security engineering practices. It allows organizations to establish confidence in their own capability, but it also helps customers to evaluate a provider's security engineering capabilities. The model is based on the idea that organizations need a repeatable, efficient and assured mechanism to improve their security engineering practices. SSE-CMM has been developed to address these needs by reducing the cost of delivering secure systems. The model contains a number of base practices which are organized into in total eleven process areas.

The Microsoft Security Development Lifecycle (SDL) [12] is another security assurance process focusing on secure software development. The purpose of SDL is to reduce the number and severity of vulnerabilities in software and it aims to guarantee security and privacy during all phases of the development process. Education, continuous process improvement, and accountability are three main concepts of SDL which emphasizes ongoing activities within the whole software development lifecycle. SDL is built upon five capability areas which correspond to different phases of the software development lifecycle, and each area consists of a collection of security activities. SDL defines four levels of maturity for these areas, namely Basic, Standardized, Advanced, and Dynamic. The basic level means little or no processes related to the activity, while dynamic level corresponds to complete compliance across an entire organization.

The Cybersecurity Capability Maturity Model (C2M2) [3] is designed to help organizations of any type and any size to evaluate and improve their cybersecurity programs. The model can be used to strengthen cybersecurity capabilities and also to prioritize actions to improve organization's cybersecurity processes. The model is organized into 10 domains and each domain has a set of cybersecurity practices. Practices in each domain will help organizations to achieve more mature capability in the domain.

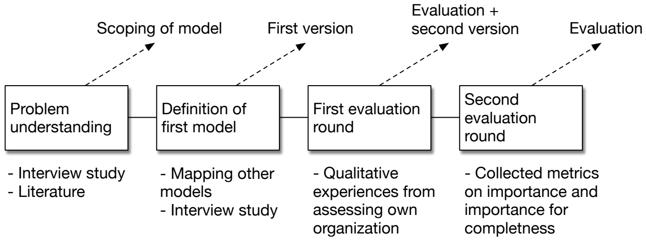


Fig. 1. Research steps

The most important features for vulnerability handling such as vulnerability identification, vulnerability assessment, vulnerability tracking and disclosure policy are included in some of mentioned maturity models. Vulnerability identification through software development process exists in BSIMM [10], SAMM [2], SDL [12] and SSE-CMM [9] and only in SMmm [1] it exits in maintenance phase. Assessing vulnerabilities only includes in SSE-CMM [9]. Vulnerability tracking by incident response team exists in almost all of them. None of them has any communication or disclosure policy except C2M2 [3]. We tried to gather all of these vulnerability handling features in our maturity model and make a complete maturity model for vulnerability handling. Being highly focused on handling vulnerabilities in third party code, our proposed maturity model should not be seen as a replacement for the other models. HAVOSS is intended to be used as a complement to other, more general, maturity models.

3 Methodology

The model has been designed iteratively based on feedback from presenting it to practitioners in the field. A first problem-understanding was achieved through an interview study with practitioners [8] where it was clear that there is a need to support these processes in industry. In that study, 8 companies in the IoT domain were interviewed about how they handle vulnerabilities in OSS and COTS code in their developed and maintained products, and what challenges they see in that. It was clear that there is a need to support these activities, meaning that the scope of the model was decided to include all activities that are relevant to identifying and solving vulnerabilities in third party (OSS and COTS) code. A literature study with a comparison to available models also showed the need for this type of model.

3.1 Research Steps

The maturity model was defined through a series of research steps as described in Fig. 1. Based on the identified need, a first version was designed. One important source was the previously conducted interview study with industrial practitioners on how they handle vulnerabilities [8]. In that study it was clear that many organizations do not have defined processes, neither for identifying, analyzing, or taking care of vulnerabilities in third party code in the products they develop and support. Another input source was already available models, as presented in Section 2. Many of the models include aspects that are related to the capability areas in our model. However, the available models are more general and not as complete in managing third party software vulnerabilities as this model. For example, BSIMM [10] includes "Software Environment" which is related to product knowledge in our model, and it includes "Configuration Management & Vulnerability Management" which is related to evaluating and remedy of vulnerabilities in our model. It is similar for the other models. They include relevant areas, but they are not as focused on vulnerability management for included third-party software where sources of vulnerabilities must be identified and monitored. Based on these input sources, a first version of the model was defined.

The model was decided to consist of *capability areas*, each consisting of a set of *practices* that can be used to identify improvement proposals in assessments. Each practice is represented as a *question* in order make it easier to interpret in an assessment. The final resulting capability areas and questions are presented in Section 4.

When the first version was available it was iteratively improved through evaluations with practitioners, in two main evaluation rounds. Helgesson et al. [6] identify three ways of evaluating maturity models when they are designed, either off-line by the authors alone, off-line by including practitioners, or on-line, i.e., in actual process improvement. Both evaluation rounds in this study can be classified as off-line by including practitioners, since all evaluations are carried out based on practitioners' opinions and experiences of trying to assess their organization. However, at this stage we have not actually conducted any improvement activities guided by the model where a before/after scenario could be analyzed.

In the first evaluation round, refinement of the model was conducted through feedback from practitioners. This was done in several sub-steps, where we in each sub-step sent the model to a contact person in industry who individually assessed their own processes with the model. When they had done that we had a meeting with the organization where we discussed general feedback on the model and we discussed a number of feedback questions, e.g. about if there were any misconceptions from researchers, if the questions were hard to answer, if there were any questions missing, and if the respondent had any thoughts about the answer alternatives. All meetings were held in a way resembling a semi-structured interview where audio was recorded, so the information could be accessed in the analysis. This type of feedback was obtained from two companies, which resulted in a number of adaptions of the model.

In the second evaluation round, feedback was received with other feedback questions than in round 1, now focusing more specifically on every practice of the model. As in the first evaluation round, the model was sent to practitioners, but in this step they were asked to consider not just the answer of each question, but they were also asked to assess the practice with respect to the following two dimensions:

– *Importance of practice*: For each question the participant was asked to judge how important the practice described by the question is in management of vulnerabilities. Possible answer alternatives were 1 – 5.
– *Importance for completeness*: For each question the participant was asked to judge how important it is to include the practice for the completeness of the questionnaire. Possible answer alternatives were 1 – 5. The given motivation was that some practices can be overlooked if they are not included in a model like this. A high score represents that the practitioner thought that it is easily overlooked if it is not included in the model. In the same way a low score means that the practice would probably be solved also without a model like this, i.e. the practice can be considered "obvious".

For each question in the model the participants were allowed to give free text comments in a separate field in the form they received.

The conducted research was influenced by design science. Compared to the framework according to Hevner et al. [7] it identified the needs and the problems in the environment e.g. through the interview study, and the evaluations were conducted with people from the same environment. The developed model was, as described above, based on available models and it represents a contribution to the available knowledge base.

3.2 Participating Companies and Practitioners

The participants in evaluation round 1 and evaluation round 2 are summarized in Table 1. The second row shows if the company participated in evaluation round 1 (✓= yes) and the third row shows how many practitioners from each company who individually answered the questions on importance and importance for completeness in evaluation round 2. The companies are working with software engineering and they represent different size, age, and maturity. Companies A, D, and F are large companies, while the other are smaller. Company E is an example of a startup while the other companies are more traditional companies. Company G offers consultancy services to other companies, while the other companies work with traditional in-house development. All companies but company C are working in the area of embedded software for IoT systems. All involved practitioners were in some way responsible for security and/or working

Table 1. Participating practitioners

Company	A	B	C	D	E	F	G
Evaluation round 1	✓	✓					
Participants in evaluation round 2	12	2	4	1	1	1	2

with security-related questions in the organization. In company A most communication was held with the main security responsible. Other persons were working within development roles.

3.3 Validity

The goal has been to obtain good validity of the research by considering typical threats to validity (e.g., [17]). Construct validity denotes to what extent the operational constructs are interpreted in the same way by the researchers and the involved practitioners. Care was taken to use as general terms as possible, not to focus on wrong specific meanings of terms in the organizations. This risk can never be ruled out completely, but it can be noticed that some terms in the model were changed in the first evaluation round, in order to not give too specific (and not completely right) meaning to the company practitioners.

Reliability denotes to what degree the analysis is dependent on the specific researchers. This is always a threat, but care has been taken to do the analysis in the whole group of researchers. The analysis has also considered feedback from members of the industrial participants. For example, both company A and B were involved in both the first end second evaluation round.

Internal validity denotes the risk that other factors than the ones known by the researchers affect the result. This is not a typical controlled study where factors are controlled, but still there may be some factors that affect the results such as ongoing and general improvement attempts with respect to security. Care has been taken to understand the situations of the participating organizations, and many of them have been involved in previous research studies with the researchers. Basically, we see the situation of the participating companies as typical examples of industrial organizations, and no major internal threats.

External validity denotes the possibility to generalize the results. All organizations are Swedish or has a Swedish origin and all participants are employed in Sweden, but they operate on an international market and most of them have mainly international customers. We do not classify them as particularly typical, but more as general examples of organizations in general, at least in the area of embedded systems and IoT systems, when it comes to their approach to managing vulnerabilities. However, in this type of study care must be taken when generalizing to other organizations.

4 Capability Areas and Practices

In this section our proposed vulnerability management maturity model is presented in detail. The six capability areas consist of in total 21 practices. In the assessment sheet, the practices are formulated as questions, e.g. A1, “Tracking maintained and used products” is formulated as “How do you keep track of which type of products are maintained and/or used?”[3]. The capability areas are

[3] The assessment sheet, together with evaluation data are available at `https://zenodo.org/record/1340623#.W2wP7RixWkB`

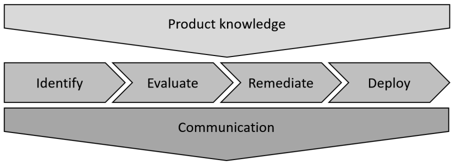


Fig. 2. The capability areas included in the proposed maturity model.

product knowledge, *identification and monitoring of sources*, *evaluating vulnerabilities*, *remedy of vulnerabilities*, *delivering updates* and *communication*. The areas and the relation between the areas is depicted in Fig. 2. Product knowledge is a prerequisite for the other areas and practices. Without this, it is not possible to efficiently, or even at all, handle vulnerabilities. Identifying, evaluating, and remediating vulnerabilities, as well as deploying updates, can be seen as areas of practices that are carried out in sequence. Finally, communication of vulnerabilities and related information can, and often should, be done in parallel with the practices and activities in the other areas. In the following subsections, each capability area and the practices are given in more detail.

4.1 Product Knowledge

Product knowledge assesses companies' knowledge of their products' components. A higher maturity level in this area indicates higher knowledge about the components. This capability area is divided into five practices:

A1. Tracking maintained and used products. Organizations should track maintained products by themselves and also products used by customers regularly, in order to be able to identify their active products.

A2. Tracking included third party OSS and COTS components included in products. Developing companies use many OSS components. This reduces the time-to-market and allow a more cost-efficient development and maintenance organization. Development is largely reduced to selecting the appropriate component to use, while maintenance is reduced to updating it when needed.

A3. Tracking used OSS or COTS versions in the included components. In addition to tracking used OSS and COTS components, it is also of importance to track the versions used in released products and firmware. Version tracking allows an efficient identification of potential vulnerabilities.

A4. Tracking possible threats that products are facing. Threats are possible dangers that might exploit a vulnerability in software products. To avoid critical dangers, and to facilitate correctness in the evaluation of vulnerabilities, it is necessary to track possible threats in software products.

A5. Specifying product usage, operating environment, and end-of-life. By specifying intended usage and operating environment, customers can better understand the intended use of a product, and it also provides important parameters when evaluating the threats and identified vulnerabilities. Specifying an end-of-life informs customers the duration for which they can expect feature and security updates for products. Note that end-of-life for feature updates and security updates can differ.

4.2 Identification and Monitoring of Sources

New vulnerabilities are found on a daily basis and there are several sources for information regarding these. A well defined and efficient process for identifying and monitoring sources of vulnerability information allow both faster and more robust management of vulnerabilities and maintenance of products and devices. The practices in this capability area focus on three aspects.

B1. Determining external sources to use for identifying new vulnerabilities. New vulnerabilities are typically recorded and identified through the CVE numbering scheme [13] and further detailed in NVD [15]. While this centralized database contains most vulnerabilities, and some other information related to them, it is also worthwhile to monitor new academic results through conference proceedings and journals, as well as monitoring security advisories and the most well-known mailing lists where software security and vulnerabilities are discussed.

B2. Receiving and following up on vulnerabilities reported to the company by external parties. In some cases, new vulnerabilities are disclosed directly to the organization. This can be the case if a third party researcher or professional analyzes the product and reports the results to the manufacturer through a responsible disclosure process.

B3. Monitoring the identified sources of vulnerabilities. Having a well defined process for monitoring vulnerability sources will help minimize the exposure time for products and devices. Often, there are exploits widely available either at the time of disclosure or very shortly after [19].

4.3 Evaluating Vulnerabilities

The goal of this capability area is to help organizations assess their maturity in evaluating the severeness and relevance of identified vulnerabilities. This has direct impact on the next area (remedy of vulnerabilities). Accurate and efficient evaluation, as well as well-founded and correct decisions regarding vulnerabilities, are prerequisites for timely and cost-efficient remediation. The practices in this area thus focus on the following two aspects.

C1. Evaluating severity and relevance of vulnerabilities. After identifying a potential vulnerability, it must be evaluated with respect to product configuration, operating environment, and threat assessment. Unused component functionality, network configuration or unrealistic exploit requirements might render the vulnerability unexploitable. Methods for ranking vulnerability severity might aid in the evaluation. A well-known metric is the Common Vulnerability Scoring System (CVSS) [5, 11], which gives a rating on the scale 0–10.

C2. Making decisions for handling and reacting to identified vulnerabilities. Firmware and software is often updated on a regular basis in order to include new functionality and patch bugs. Severe security vulnerabilities might need immediate attention and result in updates that are not within the planned cycle. Such updates are very costly and often engage large parts of the organization. It is thus very important to only perform out-of-cycle updates if necessary.

4.4 Remedy of Vulnerabilities

Based on the severity, vulnerabilities can be divided into three basic categories, namely those that need urgent changes, those that can be patched in the next planned release, and those that need no changes or updates. This capability area assesses the maturity level of organizations for handling these categories.

D1. Handling vulnerabilities that need urgent changes. Urgent changes require immediate action and will impact several processes within the organization. The organization should have an action plan for handling this event in order not to cause unnecessary and unforeseen problems.

D2. Handling vulnerabilities that are updated in a planned release. Here, the maintenance organization must make sure that the affected component is patched in the next release.

D3. Handling vulnerabilities that need no changes. When vulnerabilities have been evaluated, and the results show that attacks are impossible or very unlikely, the organization must make sure that this is well documented. If the component is not updated to a patched version, the vulnerability will always be present, so the organization must make sure that it is not unnecessarily evaluated over and over. Moreover, new information might affect the status of a vulnerability. In that case, it must be re-evaluated since updated information (e.g., new exploits), might affect the decision.

4.5 Delivering Updates

After updating the used components with the latest version, or applying patches to the software, the new firmware or updated software must be deployed to the actual devices. This does not only require a communication channel to the devices, but the channel must also be secure, including verifying the authenticity of new software. However, verifying authenticity is not enough, it is also important to make sure that updates are actually installed on devices [4]. This capability area is divided into two activities.

E1. The process of delivering and applying upgrades to deployed products. The update process can be done fully automatically if the devices support that. In some cases, users will be notified of new updates but needs to go through manual steps to apply them. In other cases, new firmware or software is posted on a website, and it is up to the user to identify and apply these patches. Exactly which process is used can be situation dependent. Although a fully automatic approach is typically preferred, requirements on system or device availability, and also privacy concerns, can render such an approach infeasible in some cases. It can be noted that a recent survey [16] based on 2205 users, reported that only 14% have ever updated their router's firmware.

E2. The process of protecting the integrity of patches while they are delivered and deployed. Integrity protection, typically through digital signatures or MACs, is needed to protect from malicious updates being installed on devices. This in turn will require a PKI or pre-shared keys.

4.6 Communication

Communicating vulnerability and security information, internally and externally, and have structured ways of doing this, allow a more robust and transparent process. It will make the security awareness more visible and contribute to more secure products. This capability is divided into six practices.

F1. Communicating internally when vulnerabilities are identified and resolved. Informing everyone within the company that is somehow affected by the vulnerability, its evaluation, remediation and deployment, allow a well-managed and structured process for updating the software.

F2. Communicating externally when vulnerabilities are identified and resolved. External communication here means e.g., producing advisories that inform the public that the vulnerability has been identified and solved. It also includes forwarding new information to other manufacturers or providing OSS patches upstream.

F3. Communicating with media when vulnerabilities are handled. Widespread and critical vulnerabilities will often come to the attention of media. Well defined processes for communicating with media can improve how the security work within the company is perceived by the public.

F4. Communicating with important customers about critical vulnerabilities. Very large and important customers might be particularly affected by some vulnerabilities, requiring a heads-up when new vulnerabilities are found. Moreover, attacks that affect important customers can have significant impact on the manufacturer's business. At the same time, such communication is resource consuming, for both parts, so it should only be practiced if necessary.

F5. Informing customers about the patching status of products. In order for customers to verify the security of their products, it should be easy to see which software, versions, and patch levels products have. This is part of what is sometimes referred to as a bill of materials. Processes for delivering such information, perhaps together with specific information related to patched vulnerabilities can ease the burden for the support team.

F6. Transferring other security related information while delivering patches. Attaching information on patched vulnerabilities and also providing information on how the patch should be applied, or which additional configuration settings should be applied, can help the customer understand why the patch is applied.

5 Maturity Levels

The intention of the maturity levels is that they should represent an increasing maturity for the assessed organization when it comes to their processes for work-

Table 2. Maturity levels used in the assessment

Level	Description
0	We don't do this.
1	We do this in an ad-hoc way based on individual's own initiatives.
2	We know how we do this, but we do it in different ways in different teams/products.
3	We have defined processes for this that are common to all teams/products.
4	We collect experience and/or metrics from our approach and base improvements on that.

ing with third-party vulnerability updates. This type of maturity can, of course, be defined in different ways, but as described in Section 2, we have chosen a way of viewing maturity that is inspired by the approach in CMMI for software development. This means that an increasing maturity implies an increased definition and standardization of approaches in the organization. We argue that this standardization is necessary in order to be able to learn from experiences and also to be effective in managing vulnerabilities. If different parts of an organization have individual responsibility to define and manage their processes for this it will not be effective. This means that we can formulate the basic contents of the levels as follows.

The first level is *level 0*, which means that no effort is spend at all on the activity. It may be that an organization does not work with vulnerabilities at all. Then they are assessed at this level. The next level, resembling level 1 in CMMI, *level 1* means that the process is carried out in some way but it is often unclear how it is done, and the responsibility is often left to developers who happen to find the need and have the right competence and resources for it. At the next level, *level 2*, there are defined approaches and routines, although there is not a standardized approach in the organization. The next level, *level 3*, represents a state where there is a standardized process in place for the practice. That is, the same, defined, procedures are used in all teams and projects. At the most advanced level, *level 4*, experiences are collected from using the standardized procedures, and these experiences are used when constantly improving the processes.

In the model presented to the participants, the maturity levels were presented as described in Table 2. When performing an assessment of the maturity, the intention is that every question is assessed. That is, there is one assessment result (level 0 – level 4) for each question. The results can then be presented either as one result for each question or a summary for each area of questions.

When improvements are identified based on an assessment it is possible to identify improvements based on the questions with low scores. When this is done there are some dependencies that can be identified. It is, as described in Section 4, possible to see that capability area A about product knowledge is a pre-condition for the other capability areas, see Fig. 2. It is therefore recommended to start with capability area A in an improvement programme.

6 Results of Evaluations

In this section the results of carrying out the evaluations are presented.

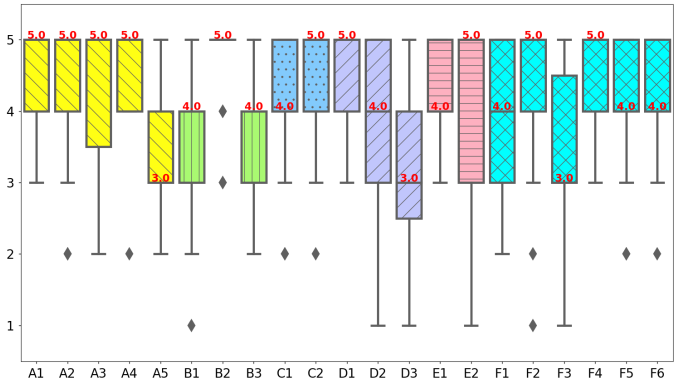


Fig. 3. Importance of activity

6.1 First Evaluation Round

In the first evaluation round a number of adaptions were made. In the discussions it was clear that the practitioners thought that there were no major misconceptions, and that the model included the major important aspects according to them. However, it was clear that some terminology that was used could be changed to terms that are more general in order to lower the risk of confusion about company specific terms. There were also some questions, especially in capability area A that were refined in order to be more understandable. Concerning the completeness, new questions about how to communicate with external sources, such as customers, were added. Also, based on the question about answer alternatives, i.e. the maturity levels, they were presented in a clearer way and the two highest levels in that version of the model were combined into the current most advanced level. In the original version there was one level for collecting experience and another level for using the experiences for improvement. These changes resulted in the model that is presented in this paper (Section 4 and Section 5).

6.2 Second Evaluation Round

In the second evaluation round the focus was on understanding the important of the questions and to what extent the questions would be handled without any model. The results with respect to *importance of activity* and *importance for completeness* of each question are shown in Fig. 3 and Fig. 4. Median values have been explicitly given to avoid ambiguity in the plots.

It can be observed that almost all questions are seen as important by the practitioners. The freetext answers reveal some more detailed perceptions. One comment on D3 (Handling vulnerabilities that need no changes) was that this might be easily overlooked. This captures the importance of the question but also indicates why it has received slightly lower score overall. It is not seen as

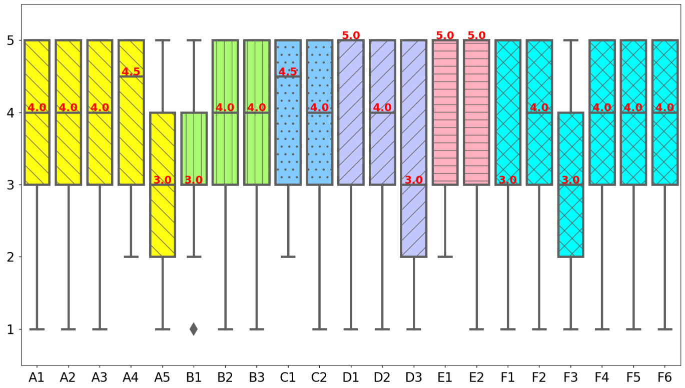
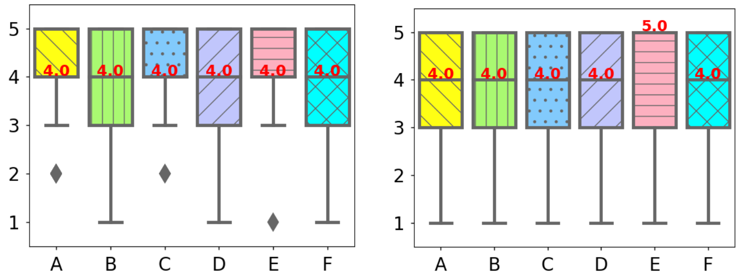


Fig. 4. Importance for completeness

important as vulnerabilities that do require changes. Question F3 (Communicating with media), which also had a relatively low score, was not present in the initial version. It was added after interviewing company A, who viewed this as an important aspect that was not covered by the other questions. One comment on this question (from another company) was that this is mostly relevant for larger companies.

Some freetext answers also suggested adding more questions. One suggestion was to add security assessment, in which assets are identified. The importance of such a question will depend on to which extent the company knows which assets are actually protected. Another suggestion was to also consider how third party components are selected. Components, and in particular their maintainers must be trusted not to e.g., add malicious code into the software. To see if there are



a) Importance b) Importance for completeness

Fig. 5. Grouped results

differences between the capability areas, we aggregate the answers to these areas, see Fig. 5. Again, it can be seen that the values are high, and there are only minor differences between the areas. A Kruskal-Wallis test (non-parametric alternative to one factor, n-levels, ANOVA) shows that no significant difference could be found between the areas ($p = 0.23$ for importance), which can be expected from the graphs.

Approximately half of the evaluation answers were from company A. We compared the results from company A to the results from the other companies by looking at box-plots and it seems they are not different. This is a motivation why we analyze the results from every respondent without considering the company. These differences were also analyzed for each question with Mann-Whitney tests, but as expected no significant differences were found. Company A was shown a summary of their responses and asked if they think their result would have differed 2 years back or 2 years from now.

> Not significantly different. If we look back a bit further, say 5 years, most activities have definitely increased in importance. Some activities will probably increase a bit further in the future, especially the F-section where laws and regulations might play a part, but overall the activities are already perceived as important. Reduced importance is unlikely in the foreseeable future.

They were further asked if the similarity between Company A and other companies were expected.

> It's expected. It indicates the increased attention to security issues is not restricted to specific businesses and this is what we have perceived as well.

That is, it can be seen that company A are working with improvements that are in line with the model.

7 Conclusions

The presented maturity model aims to help organization assess their maturity in handling software vulnerabilities in third party OSS and COTS components. The importance of such a model is due to the increasing number of vulnerabilities that are being reported, and the growing number of connected devices that are bound to change the society in the near future. The model is based on six capability areas and 21 practices. Related maturity models, i.e., those that focus on software security are very broad and cover many aspects related both to software development, maintenance, and organizational aspects, but they are not detailed enough to cover all aspects of handling vulnerabilities in third party code. Thus, this model can be seen as an important complement to other well-known models. This is also supported by our evaluation, which shows that the defined practices are highly relevant.

Acknowledgements This work was supported partly by the Wallenberg AI, Autonomous Systems and Software Program (WASP) and partly by Swedish Governmental Agency for Innovation Systems (Vinnova), grant 2016-00603.

References

1. April, A., Hayes, J.H., Abran, A., Dumke, R.: Software maintenance maturity model (SMmm): the software maintenance process model. Journal of Software: Evolution and Process **17**(3), 197–223 (2005)
2. Chandra, P.: Software assurance maturity model - a guide to building security into software development. Tech. rep., OWASP (2017)
3. Christopher, J.D.: Cybersecurity capability maturity model (C2M2). Tech. rep., Rhodes University (2014)
4. Cui, A., Costello, M., Stolfo, S.J.: When firmware modifications attack: A case study of embedded exploitation. In: Network & Distributed System Security Symposium (2013)
5. FIRST: Common vulnerability scoring system v3.0: Specification document, https://www.first.org/cvss/specification-document, Last accessed 2018-06-03
6. Helgesson, Y.L., Höst, M., Weyns, K.: A review of methods for evaluation of maturity models for process improvement. Journal of Software Maintenance and Evolution: research and Practice **24**, 436–453 (2011)
7. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly **28**(1), 75–105 (2004)
8. Höst, M., Sönnerup, J., Hell, M., Olsson, T.: Industrial practices in security vulnerability management for iot systems – an interview study. In: Proceedings of Software Engineering Research and Practice (SERP) (2018)
9. ISO/IEC: Information technology — security techniques — systems security engineering — capability maturity model. Tech. rep., International Organization of Standardization (2008)
10. McGraw, G., Migues, S., West, J.: Software security and the building security in maturity model (BSIMM). Journal of Computing Sciences in Colleges **30**(3), 7–8 (2015)
11. Mell, P., Scarfone, K., Romanosky, S.: A complete guide to the common vulnerability scoring system version 2.0. In: Published by FIRST-Forum of Incident Response and Security Teams. vol. 1, p. 23 (2007)
12. Microsoft: Simplified implementation of the Microsoft SDL. Tech. rep., Microsoft Coporation (2010)
13. Mitre: Common vulnerabilities and exposures. https://cve.mitre.org/, (visited on: 2018-05-15)
14. Niazi, M., Wilson, D., Zowghi, D.: A maturity model for the implementation of software process improvement. Journal of systems and software **74**(2), 155–172 (2005)
15. NIST: National vulnerability database. https://nvd.nist.gov/, (visited on: 2018-05-15)
16. Powell, M.: Wi-fi router security knowledge gap putting devices and private data at risk in UK homes. Tech. rep. (2018), available at https://www.broadbandgenie.co.uk/blog/20180409-wifi-router-security-survey
17. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering **14**, 131–164 (2009)
18. SEI: Capability Maturity Model Integration, Version 1.2, vol. CMU/SEI-2006-TR-008(2008). Carnegie Mellon Software Engineering Institute (2008)
19. Shahzad, M., Shafiq, M.Z., Liu, A.X.: A large scale exploratory analysis of software vulnerability life cycles. In: Proceedings of International Conference on Software Engineering (ICSE). pp. 771–781 (2012)