Tool Support for Security Classification for Internet of Things (long version)

Manish Shrestha , Christian Johansen , Maunya Doroudi Moghadam , Johanna Johansen , Josef Noll

Research report 495, September 2020

ISBN 978-82-7368-460-8
ISSN 0806-3036

Tool Support for Security Classification for Internet of Things (long version)

Manish Shrestha Christian Johansen Maunya Doroudi Moghadam
Johanna Johansen Josef Noll

September 2020

Abstract

DevSecOps is the extension of DevOps with security aspects and tools throughout all the stages of the software development life cycle. DevOps has become a popular way of developing modern software, especially in the Internet of Things arena, due to its focus on rapid development, with short cycles, involving the user/client very closely. Security classification methods, on the other hand, are heavy and slow processes that require high expertise in security, the same as in other similar areas like risk analysis or certification. As such, security classifications are not compatible with the DevSecOps, which primarily goes away from the traditional white-hat hacker team style of penetration testing that is done only when the software product is in the final stages or already deployed.

In this work, we first identify five requirements for a security classification to be *DevOps-ready*, two of which are the focus for the rest of the report, namely to be tool-based and easy to use for non-security experts, like ordinary developers or system architects. We then proceed to exemplify how one can make a security classification methodology DevOps-ready. We do this through a prototyping process, where we create and evaluate the usability of a tool supporting (or implementing) the chosen methodology. Such work seems to be new within the usable security community, let alone in the software development (DevOps) community. Therefore, we present our process as a recipe that others can follow when making DevOps-ready their own security methodologies, which we believe to be valuable since it would both make the methodology more user friendly for themselves at the same time as widening the range of population that can take in using their methodology. The tool that we built is more of a byproduct contribution of the above, even though it can be independently used, extended, and/or integrated by developer teams into their DevSecOps processes, most probably during the testing phase where the security class would be one of the metrics used to evaluate the quality of their software.

[0] *Address for correspondence:*
Department of Technology Systems, University of Oslo, P.O. Box 70, 1027 Kjeller, Norway.
E-mail: `cristi@ifi.uio.no`

Contents

1 Introduction

According to International Data Corporation, the predicted number of Internet of Things (IoT) devices for 2025 is 41.6 billion, generating about 7.9 zettabytes of data[1]. Because of this amount of produced data and human life penetration (e.g., in smart homes, offices, cities, hospitals), it is highly essential to develop secure IoT systems. However, securing IoT still proves challenging, especially in industries focused on functionality and low costs demanded by the high competition on the market, as argued by, e.g., [24, 14, 17].

IoT software, like most modern software, are developed in an agile style (see e.g., the Scrum[2] method), where popular now is the DevOps culture [8]. One important mantra of Agile[3] is to include the user (or the client) of the software at all stages of the development in a continuous manner. One reason coming from the developers is that in this way, the client/user will be more acquainted with how the software is being built and will tolerate more the bugs and downsides of each version. DevSecOps[4] adds security tools and awareness at all stages of the software development life-cycle [12]. However, the security tools [15, Part VI] that can be adopted need to have a low threshold in terms of learning and usability so to be able to be effectively included in the DevOps tool-chain [9].

Security is traditionally considered by the industry as an aftermath, a non-functional requirement that needs experts, e.g., white-hat penetration testing teams, to evaluate. Traditional methods like certification, security classification, or risk analysis cannot keep up with the changing threat landscape in IoT systems [19]. Standards such as ISO 27001 and certification such as Common Criteria are long and document-oriented processes. Keeping up with the software changes in short and frequent release cycles as in agile means updating the required documents regularly, which is not feasible. Similarly, labelling schemes such as UL Security Rating [16] or BSI Kitemark[5] are mostly based on penetration testing and risk analysis, besides documentation. Risk assessment methods require significant amounts of time and resources to conduct. Examples of risk assessment frameworks include CORAS [11], EBIOS[6], TVRA [7], FAIR [13], and OCTAVE[1]. The result of conducting a risk assessment on a system at least provides a good overview of the critical components and security threats for the system. However, these approaches follow a waterfall model where the assessments are not frequent as compared to the releases, and thus may not fit the agile style of system development [10].

As such, the software industry (and especially the IoT one) lacks motivation (i.e., when the difficulty is high the motivation too needs to be high) and lacks guidelines for building security by design. We think that DevSecOps is one positive contribution in this respect since it aims to lower the threshold for security aspects (e.g., tools, procedures, methods, guides) to enter the development process.

Security classification methods are not easy to integrate into the DevSecOps, and even more so for IoT [5] where regulations, guidelines, and frameworks are only recently starting to appear (see e.g., IoT Security Foundation (IoTSF)[8], Global System for Mobile communication

[1]https://www.idc.com/getdoc.jsp?containerId=prUS45213219

[2]ScrumGuides.org

[3]http://agilemanifesto.org/principles.html

[4]https://www.devsecops.org

[5]https://www.bsigroup.com/en-GB/about-bsi/media-centre/press-releases/2018/may/bsi-launches-kitemark-for-internet-of-things-devices/

[6]https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/rm-ra-methods/m_ebios.html

[7]https://www.etsi.org/deliver/etsi_ts/102100_102199/10216501/05.02.03_60/ts_10216501v050203p.pdf

[8]https://www.iotsecurityfoundation.org/wp-content/uploads/2018/12/IoTSF-IoT-Security-Compliance-Framework-Release-2.0-December-2018.pdf

Association (GSMA)[9], IoT Working Group of the Cloud Security Alliance (CSA)[10], or the Industrial Internet Consortium[11]).

What we do in this work is to first identify, based on our experience with security classifications and on investigating (online) literature about DevOps tool-chains and practices, five principles (or requirements) for a security classification to be DevOps-ready. In short, these are: (1) dynamicity, (2) tool-based, (3) easy to use, (4) static impact, and (5) oriented on protection mechanisms (detailed in Section 2.2). We then choose an existing security classification methodology from [22] that already satisfies (4) and (5) and focus here on making it satisfy the two requirements (2) and (3). Since the first requirement is dependent on (2), we do not consider it here.

We are thus developing a tool, implementing the chosen methodology, and testing its usability on users selected to represent well our target group, i.e., non-security experts such as software developers, designers, architects, IT managers, or personnel from software operations. Our users described more thoroughly in Section 3, are: (i) partners from one large European IoT project and students from one course on IoT security, both of which we involve several times during several stages of the development; as well as (ii) SMEs from a Polish Cluster (involved only for evaluating a preliminary web-based version of the tool) and (iii) several developers recruited from the industry (i.e., from software developing companies) with whom we test the final version of the tool. Due to the nature of our process, we have mainly used workshops and interviews as our methods to evaluate our prototypes and to extract information from our users. We also used online questionnaires and UX logging, though with not so much inputs as the workshops.

We do our work in four stages, developing three prototypes along the way; this is what we describe in Section 4 (the manual stages) and Section 5 (the tool prototypes). We present this part as a "recipe" to make it easy for others to transform other security classification (or similar) methods into DevOps-ready tools, by following and maybe adapting our stages and "ingredients". We have worked on purpose to make these stages intuitive and natural, following interaction design principles, but applied to this peculiar task, i.e., taking a complex, expert-oriented, method and transforming it into a tool that can be used by not-so-experts. In short, one first needs to evaluate (see Section 4.2) the chosen security methodology as it is described in available documents or by experts; in our case, the methodology also had examples of applications to SHEMS (Smart Home Energy Management Systems) [20] and AMI (Advanced Meeting Infrastructure) [22]. Then one needs to transform the methodology into a process (steps to follow) focused on the non-expert target users (see Section 4.3). The process then should be implemented into a tool, albeit a very simplistic tool, like in our case using spreadsheets (see Section 5.1), so to test the automation and procedure flavour of the method. From the evaluation of this simple first implementation, one can draw more concrete requirements for the actual tool to be implemented (see Section 5.2). Then one sets to implement and evaluate version of this tool until a stable variant is reached (see Section 5.3) that can be a candidate for integration into a DevOps tool-chain.

We are currently working with the software company eSmart Systems AS that provides cloud-based solutions for smart grid monitoring of AMI to take up into their development process the tool that we present in this technical report. From this point on we do not see significant research challenges, but only technical integration and maybe more iterations of UX adjustments/improvements to fit the actual development process of this software company.

[9]https://www.gsma.com/iot/iot-security-assessment/

[10]https://downloads.cloudsecurityalliance.org/assets/research/internet-of-things/future-proofing-the-connected-world.pdf

[11]https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB.pdf

2 Security Classification for DevSecOps

2.1 DevSecOps and Usability of Security

Traditional software development life-cycle can be presented at a high-level, using the waterfall model (see e.g., [18]). Here, the software development goes through the stages of (i) requirements definition, (ii) software design, (iii) implementation, (iv) testing, and (v) maintenance; similarly to factory production line processes where one team works in one stage and when finished with their artefact, hand it over to the next team to start their stage. Even though such development styles are suitable when methods and techniques are known, and designs and requirements are of high importance like in large scale projects, it has become obsolete in many areas, especially for SMEs and small projects as in IoT.

Instead, agile methods [6] have become popular, which take from the spiral model [4] a cyclic way of developing software, revisiting the same stage multiple times, e.g., requirements might change, or new requirements introduced because the client or the market dictate it. When looking at their manifesto[12], the agile style of development seems a radical change in software development culture because agile methods value: (i) individuals and interactions over processes and tools; (ii) working software over comprehensive documentation; (iii) customer collaboration over contract negotiation; (iv) responding to change over following a plan. It is clear that agile methods promote more the inclusion of users, as advocated by the interaction design community. However, agile is only a style of development, a philosophy or culture change, and thus is not always clear how to implement and often left to the understanding of the CTO. The Scrum method is popular probably because the ones that introduced it were very comprehensive in their recommendations[13], making it easier for companies to implement.

DevOps can be seen as an agile method that differentiates itself through the fact that it is open to and encourages the use of tools at all stages, including the operations stage (thus the 'Ops' in the name). Operations have become more important lately, not only because of the proliferation of the cloud, making the infrastructure cheaper to deploy and run the software, but also because of automation and tools becoming available for more tasks in all the development stages. DevSecOps more recently brings into the DevOps the security, following the same philosophy, i.e., security awareness (or best practices) and security tools/processes at all stages. In particular, the penetration testing that depends on a high level of security expertise (usually coming from outside the team) is mostly replaced by security tools such as code scanners, loggers, or API security testing, and stage relevant security education for all team members.

We see DevSecOps as an arena that promotes the industrial adoption of usable security tools more than ever. On the one hand, since DevSecOps is tool intensive and lowers the usability threshold allowing more (and less usable) tools to be incorporated into the development tool-chain. On the other hand, DevSecOps is so open to new tools that offer researchers a motivation to make the security tool easier to use, hoping that is will be adopted by the industry.

2.2 Principles for DevOps-ready Security Classifications

We have identified five general *principles/requirements for making a security classification DevOps-ready*, by which we mean that the security classification can be easily integrated into a DevSecOps tool-chain as one of the security mechanisms/tools for developing quick and secure (IoT) systems. These principles can easily be applicable to similar other expertise-heavy methods like risk analysis (which are usually manual, slow, and expensive [2, 23]).

[12]http://agilemanifesto.org/principles.html

[13]https://scrumguides.org

The reader acquainted with security classifications might find the text below easy to follow. However, someone else might have difficulties with some of the (albeit succinct) arguments behind the five principles, but we trust that after going through the details of Section 4.1, the ideas presented below will be easier to appreciate. For now, we are contented to give a brief definition of what we understand a security classification to be (in very general terms).

> A *Security Classification Methodology* (SCM) has the goal to evaluate the security of a system with the outcome of classifying it, thus a security class offering a measure of the strength of the system. SCM (s.a. the ones from the French agency ANSSI or the US agency NIST) are often used for governmental systems, whereas similar methods for risk assessment (s.a. the standard ISO/IEC 27005 or the EBIOS from the European agency ENISA) are more often used by industry, and involve more calculations of losses and countermeasures in case of breaches. SCM compute a *security class* by combining evaluations for *Impacts* and *Likelihoods* (in case the system is breached), where the likelihood is the result of combining the evaluations of the *Exposure*, the users' *Accessibility* to the system, and the power of *Attackers*. Exposure, in turn, is determined by combining the *Connectivity* and the security *Protection* mechanisms supported by the system.

Based on our experiences with security classifications and with DevOps development practices, we consider the following principles as a minimum for a DevOps team to be able to adopt a new security classification methodology.

1. **Dynamic.** In evergreen[14] applications, which are nowadays popular like with web browsers[15], the development never ends, and updates (both functional and security/bugs patches) are constantly pushed to the deployed system, preferably without user interaction (e.g., consent). Therefore, any security classification needs to be dynamic so that it can be reevaluated for each update; similar to how software testing is being done. The dynamicity implies that the evaluation needs to be performed quickly to cope with the short development life-cycles of DevOps.

2. **Tool-based.** The method has to have a tool support, and necessarily not only with a GUI but also with a REST/API available so that is can be integrated within the overall DevSecOps tool-chain (e.g., [9]). Tools nowadays built with UI (like web-based apps) are also built with an API to which the UI connects, so the API requirement is not difficult to have as a byproduct of the tool-support requirement.

3. **Easy to use for non-security experts.** This is an essential requirement, allowing a security tool to be taken up into a DevSecOps framework because one of the main goals of DevSecOps is to move away from the traditional style of white-hat penetration teams who evaluate the security of a ready-built (or deployed) system, and into a new style where every member of the DevOps team needs to have security competence relevant for their field of development. Thus, a security classification method for DevSecOps needs to be usable by non-security experts, who otherwise know much about the system under development (e.g., developers or system architects).

4. **Impact statically and manually evaluated.** Security classifications (the same as risk analysis methods) involve evaluating the impacts of security breaches (or attacks). However, to use the security classification inside one company for developing one product,

[14]https://www.danielengberg.com/what-is-evergreen-it-approach/
[15]https://www.techopedia.com/definition/31094/evergreen-browser

the impact evaluation is nearly static because the planned product and its functionalities and applications do not change (at least not outside the first development phases) almost throughout the lifetime of the product. As such, the security methodology is enough to evaluate impacts once, in the beginning (maybe using even security experts), and input this evaluation manually to the tool. Therefore, we assume that impacts are of no concern for the rest of these requirements.

5. **Fine-grained security functionality oriented.** Outside impact, security classifications are usually attack-centric, focusing on the capabilities of the attackers. For IoT and for DevOps style of development, we want to focus less on attackers, which are very dynamic and difficult to evaluate, and more on the security protection functionalities and exposures of the system under development. Focusing on functionalities makes it easy to evaluate the system within a DevOps testing cycle automatically, and also allows the developers to understand how to make their systems secure by design by indicating which functionalities are a good match for which exposures and with what protection level (derived from the class specifications).

The methodology that we work with is already developed to meet requirements 4 and 5. Thus we do not evaluate these here. Moreover, the dynamicity (i.e., requirement 1) can be achieved and evaluated only after a tool is built. Therefore, in this work, we focus on the two requirements, 2 and 3.

3 Users

For this work, we had access to the following users for testing our prototypes:

SCOTT project. The most inputs and interactions were done with the participants from one large project called Secure Connected Trustable Things[16] (SCOTT) with 57 partners from industry and academia from 12 countries working on ca. 15 pilots involving ca. 30 IoT technological building blocks.

Students. They were the participants in one course on IoT. There were relatively few student participants, but their inputs were valuable and representative for their target group.

SME cluster. Through a 'hackathon' we reached out to a cluster of SMEs (Small and Medium-sized Enterprises) doing technology development from Poland.

Software experts. Besides the above subjects, we also reached out to four individual participants from the industry who have long software development experience. The background of the participants are described below:

- Participant 1: CEO of a startup company with more than 25 years of experience in the software industry, especially software used in the energy sector. His experience includes management and training, software design, development, and testing.
- Participant 2: CTO of another company with more than 20 years of experience in the software industry, also having a good background in information security.
- Participant 3: Senior Consultant and Business Developer in another company with more than 20 years of experience in software development.
- Participant 4: Software engineer with ca. 7 years of experience, having worked as a software engineer and data scientist in several companies.

[16]https://scottproject.eu

The target groups that we consider are motivated by Principle 3 from Section 2.2, and in short, these should focus on *non-security experts*. More precisely, we are interested in people that have technical expertise, especially for our current study those having IoT technology knowledge, but also more generally, people like system designers and developers who are not security engineers but who may have some basic security training (since their routine tasks need this) but maybe specific for their particular area of expertise. We are also interested in non-technology experts, like CEOs and managers of various development and operations aspects of technology development; these people would know about use-cases, features, or economy and impacts, related to the technology system, but maybe not about the technical details.

Particularly, the SCOTT project participants were usually teams made of both technical and management people, and on rare occasions, a person with considerable security expertise. The 'Software experts' category is, similarly, made of high-expertise people. More to the contrary, the 'Students' are still technical people, with little knowledge of security and fresh in the development field also. The 'SME cluster' is supposed to have teams that are most diverse in expertise, from business experts to developers, but not much security.

We explain in the rest of the technical report, how and for which of our studies we interacted with the different users from above, to test the usability of the security classification methodology and of the tool that we present in this technical report.

4 Manual Security Classification

4.1 Reviewing the Security Classification Methodology

The security classification methodology that we take as the starting point in this work has been proposed in [22] as an extension of the standard for "Security Classification of Complex Systems" developed by the French national agency ANSSI. Besides, the methodology of [22] incorporates (and conforms with) security concepts from several other relevant standards from among others ISO/IEC, ETSI, OWASP, ENISA. This method has been detailed and extended towards IoT systems in [21].

Terminology: We will often abbreviate Security Classification as SC, and when we refer to SC Methodology we will use SCM, whereas for the SC Tool presented in the rest of this report we use SCT, maybe with versions attached as SCTv1 if we want to emphasis the different version that the tool prototype went through.

In short, the methodology is based on the analysis of impacts, connectivity, and protection level of the system. Protection level is determined from the protection mechanisms that are applied to the system. Protection level combined with connectivity forms the exposure level, and finally, exposure and impact are used to determine the security class of the system, as displayed in Figure 1. SCM considers five levels of Connectivity [21, Sec.3.1] adopted from ANSSI.

The protection mechanisms are evaluated based on a list of security criteria [20, Table 3] that sum up to a protection level (from P1 to P5). The higher the protection level, the more security mechanisms it includes (when relevant, e.g., for the connectivity of the system). Finally, the classification methodology considers five impact levels also taken from ANSSI (see [20, Sec.3.7]), namely Insignificant, Minor, Moderate, Major and Catastrophic. The impact level is determined usually by security experts.

A lookup table is used to determine the exposure from connectivity and protection levels, as shown in Table 1. Finally, the security class is determined from the exposure and impact using a class lookup Table 2.

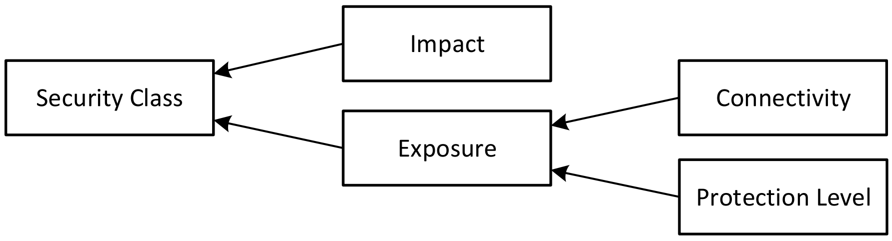


Figure 1: Components of the evaluation of a security class.

Table 1: Calculations of Exposure Levels

P1	E4	E4	E5	E5	E5
P2	E3	E4	E4	E5	E5
P3	E2	E3	E3	E4	E4
P4	E1	E1	E2	E2	E3
P5	E1	E1	E1	E1	E2
Protection/ Connectivity	C1	C2	C3	C4	C5

Table 2: Calculations of Security Classes

Catastrophic	A	C	E	F	F
Major	A	B	D	E	F
Moderate	A	B	C	E	E
Minor	A	A	B	D	D
Insignificant	A	A	A	C	C
Impact/ Exposure	E1	E2	E3	E4	E5

4.2 SC Methodology Evaluation

The development of a Security Classification Tool (SCT) involved multiple stages of prototyping and usability testing, as described below.

The very first stage, however, was to take the methodology as described in the research papers and evaluate the usability claim, i.e., that the method is easy-to-use for non-experts in security. For this evaluation stage, we interacted only with two of our user groups, namely with the students and SCOTT partners (which included companies such as Philips Research[17] (NL), Vemco[18] (PL), AVL[19] (AT), ISEP[20] (PT), VTT[21] (FI) or Tellu IoT[22] (NO), as well as academics, e.g., from Gdansk University of Technology[23]).

Our research team includes security experts, and thus we first read relevant papers and understood from [22] the SCM ourselves. We then prepared a presentation for the two groups of users. To the SCOTT partners, we presented and explained the SCM through several short workshops (30min to 1h). The participants from the SCOTT partners were a mixture of technology people, with management and software/system design people; however, there were no security experts in their teams, except for some of the technology people who had general security knowledge or specific for their technical field. To the students, we presented the SCM shortly in one of the lectures from the beginning of the course and gave as a homework, the methodology papers which they were supposed to apply to their IoT system exercise (recall that the course was on IoT systems and security).

The first results can be summarised as rather discouraging for the SCM. Although the participants did express interest in the concept of security classes, none of them could understand much from the SCM, let alone how to apply it to their use cases. This was one major observa-

[17]https://www.philips.com/a-w/research/home
[18]https://vemco.pl/
[19]https://www.avl.com
[20]https://www.isep.ipp.pt
[21]https://www.vttresearch.com/en
[22]https://www.tellucloud.com/
[23]https://eti.pg.edu.pl

tion that we collected from interactions during the workshops. We did not obtain more concrete suggestions, mainly because the participants could not understand enough about SCM to give us meaningful comments. (We can also be understanding towards this outcome, since students are shy in giving critiques, and technical people are usually careful to giving suggestions if they do not understand the technology presented.)

Our team then took another attempt at simplifying the presentation, and more importantly, we now presented how the SCM would be applied, focusing on the application to SHEMS published in [20].

Each SCOTT partner was involved in one or more of the ca. 15 pilots of the project, all developing IoT systems, e.g., Philips was coordinating a pilot on “Assisted living and community care systems”, Vemco was coordinating a pilot on “Secure Connected Facilities Management”, whereas GUT was involved in both of these pilots, and VTT was coordinating a pilot on “Air Quality Monitoring for Healthy Indoor Environments”. We reasoned that by presenting an application of SCM to a similar IoT system, they would easily understand how to apply the SCM to their use case. We also took the energy to read through the various project documents where their respective IoT systems were being described (preliminary versions, since we were in the middle of the project). We then tried in our presentation to make some (rather superficial) correlations between the application of the SCM to the SHEMS and to their respective pilot systems. For the students, we could not do this second iteration.

This second presentation did not manage to clarify enough as to allow the participants to apply the SCM. However, we did get more interactions during this second round of workshops. Several discussions were held in the form of question and answer, directed from the participants to us, the presenters. The topics included some of the details of the SCM, like the calculation of impact, or the evaluation of connectivity. One major outcome emerged at the end, where the participants endorsed, rather unanimously, the observation of one of them, which was

> “It is not clear where to start with this methodology”.

This observation becomes quite evident when thinking more about it, e.g., certification bodies use certification processes to do their work. The most simple definition of a ’process’ implies a sequence of steps to be followed to arrive at a desired outcome. In our case, this meant we needed to produce a sequence of steps that a non-security expert could follow in order to evaluate the security class that a system belongs to.

4.3 SC Methodology as a Process

Based on the feedback from the evaluation stage (e.g., involving questions/observations related to what kind of information about the IoT System were needed), we expressed the security classification methodology as a ten steps process as follows:

1. **Define the IoT system.** The user decides which system should be evaluated and gathers knowledge about the system s.a.: system architecture, functionalities, security requirements, use cases, and context of use. This step helps the user to understand at a high level, and prepare, the system under evaluation.

2. **Define the components of the system.** A system is composed of one or more components. In this step, the necessary components of the system are defined. Examples of components for a smart home are IoT hub, smart devices, sensors, control data, etc.

3. **Describe the features of system components.** The interactions between the system components are now described. The user decides on the use case where the security classification should be applied. At this point, the user already has a reference architecture of the system.

4. **Define the impact level.** For each component, the worst impact of security breaches is defined. The levels of impacts are defined by the SCM as Insignificant, Minor, Moderate, Major, and Catastrophic (the same as ANSSI does). This step is similar to the evaluation of impact risk assessments. The impact may be on the economy, human life, physical infrastructure, business, etc.

5. **Describe communication mechanisms.** The communication capabilities for each component are described. The user will look into which communication standards are used.

6. **Describe the type of networking.** The user has to find out whether the network is only a Home Area Network or a Wide Area Network.

7. **Determine the Connectivity Level.** Based on the two previous steps, the user assigns the connectivity level to the components. The connectivity level varies from C1 to C5 and is described by the SCM.

8. **Determine the protection Level.** Relevant security criteria are defined for the components, and the security functionalities they have are also listed. The list of protection criteria and security functionalities obtained is compared to the Protection Level table given by the SCM, to determine to which protection level the existing security mechanisms belong to.

9. **Determine the exposure level.** Protection level and connectivity determined in the previous steps are used to identify the exposure level using a lookup table defined by the SCM.

10. **Determine the security class.** The security class is now determined using the exposure and impact levels based on the class lookup table defined by the SCM.

Working with the SC methodology is manual, as far as the research papers [22, 21] describe it. Therefore, the above process is also manual, with the advantage being that a clear procedure is given to the user to follow, besides the research papers. One can easily see in the above steps that some can be more or less automated. Automation is a highly desired method of making a difficult technical process more user friendly, since when compelled to use technology, there is nothing better than not using it. Steps 1 to 3 are manual, and the user can take as much time and space for writing down the description as required (no page limits). Step 4 is a classical risk analysis stage which we assume to be more static for DevOps and IoT software systems. It is also manual and requires security expertise (depending on the company's internal desires for strictness with the evaluation, since the impact is something that cannot be changed much by developers, as opposed to protection measures and connectivity functionalities). Step 5 and 6 are also manual and needed only to help in step 7. Step 8 is probably the most tedious because of the long list of criteria that need to be evaluated. However, the list helps the user to assess the security thoroughly. Steps 9 and 10 are done through lookup tables.

As such, it can be seen that steps 9 and 10 can easily be automated, whereas steps 1 to 7 not so easily, at least the SCM does not give us any help in that direction. Step 8 can be partly automated by summing up all the answers of the user and comparing them automatically with the respective table from the SCM.

4.3.1 Evaluation of the ten-steps process

Designing and evaluating the ten-step process for the SCM was done over several workshops (each of 30min to 1h) interacting with the SCOTT users only. One significant activity during this stage was to apply the SCM ten-steps to the pilots from SCOTT together with the respective partners. We made two applications, to:

1. the “Elderly UI” component of the “Assisted Living and Community Care System” (AL-CCS) pilot, and interacting mostly with Philips (who were coordinating this pilot) and other technical people that were closely involved in the team developing this ’care-at-home’ system;

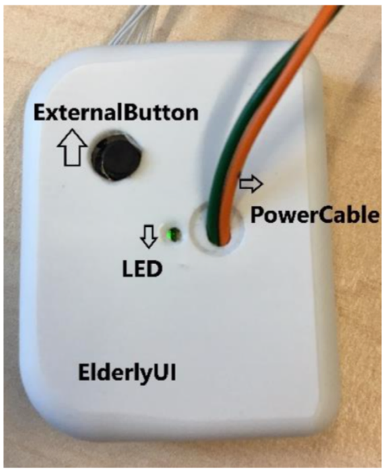


Figure 2: Early prototype of the ElderlyUI component.
(Description and image courtesy of Philips Research.)

2. the “Multimodal Positioning System” (MPS) component of the “Secure Connected Facilities Management” pilot, and interacting mostly with Vemco (who were coordinating this pilot) and other technical people from Gdansk.

Our work was based on reading the respective documents from the project and interacting with the team building the respective system.

In short, the Elderly UI (see Figure 2) is a small form factor prototype device that can be worn as a patch on the skin for weeks at a time without the need for recharging and can continuously observe activity and position from the elderly resident, and periodically transmits the observations straight to the cloud. The MPS had as main functionality the localisation of people and assets within critical infrastructures, being applied in this case inside a refinery. For our work on applying the SCM, we have used the technical project-internal documents for each system to collect the necessary information for evaluating the connectivity, protection and exposure levels. Then during the workshops, we adjusted our understanding of the system and worked with the teams to properly apply the SCM to their system. The ten-steps methodology went through two major redesigns, where mainly the order and the number of steps were changed, and the helping descriptions were improved.

During these workshop interactions, we had two goals:

1. Us to understand the IoT system of the SCOTT pilot that we wanted to use as application and to understand better how the ten-steps process worked and how easy it was to apply;

2. The SCOTT users to understand how the SCM works and how to use it to apply it themselves.

For both goals, our interactions were geared towards collecting observations about the usability of the ten-steps and how to refine it to fit the two examples that we considered as the representative of the general application area that the SCM was intended for.

One playful activity that our users seemed to enjoy was to work on identifying how the security class can be improved; e.g., for the ElderlyUI system, we had scenarios that changed the class from E to B by making changes and updates to the system. This is one major benefit of the security classification methodology that is claimed by the main article [22]. Therefore, our interactions seem to confirm this claim that IoT developers would enjoy knowing the security class of their system, which in turn would encourage them to strive to improve their system's security so to improve the class.

4.3.2 Outcomes and Major Observations

Besides the constant feedback that we received during the workshops about small improvements to the ten-steps process, we drew the following major observations.

1. The participants could answer most of the ten steps questions when we were guiding them. The guiding meant us, e.g., explaining the purpose of a step (often mostly confirming that their understanding of that step was matching with ours); or giving more details about a step like what was meant by the Home Area Network (HAN).

2. The most difficult parts of the methodology were identified as being:

   (a) the evaluation of the Impact level, which looked to them like a job for security experts doing risk assessment (which the participants were not); and

   (b) finding the Protection level since it involved answering many specific security questions which needed interactions with other members of their development teams (i.e., those that worked on the respective aspect that the question in our table referred to).

However, the SCM papers [22, 20] especially point out that the evaluation of the impact level is not a specific concern of the SCM and is supposed to be similar to how risk assessment or similar methods evaluate impacts of attacks. Moreover, the impact level is only indicative and does not need to be done to a perfect detail for one to use the SCM as it was intended. Therefore, we could not do anything about the first observation; and it was not our research goal to do so anyway, since we were taking the SCM as given, and not as something to improve as a security instrument per se. Our goal, as one can recall from the Introduction, is to take a security classification methodology as it is, and make it DevOps-ready by building a tool that makes it easy for non-security expert users to apply it.

The second observation is directed to a core aspect of the chosen SCM, since the list of security functionalities that the observation refers to, is a main differentiating aspect claimed by [22, 20]. Therefore, we decided to improve on how the users work with this list in the next iteration of the tool, which was now decided to be computer-based.

At this point, we were also ready to test the ten-steps with more users, but it was decided together with the SCOTT users that an online tool would be best suited for allowing more users to join our testing sessions.

5 Interaction Design Tool Development Process

5.1 Spreadsheet implementation

Based on the feedback from the users, we then implemented the ten-steps manual process from Section 4.3 into a tool based on spreadsheets. As much as this first implementation can be called so, we consider that a *'tool'* is something run by a computer to help the user with a specific task by organising, guiding, and maybe automating some of the aspects of the task. In our case, the process of security classification was the task at hand, which also had some of the steps ready for automation; whereas for the other steps the tool should be seen useful only to organise the work and gather inputs from the users.

The spreadsheet tool was implemented in Google Sheets because it is a cloud-based application where a team can collaborate in real-time. Figure 3 shows a snapshot of the spreadsheet-based SCT. Our goals were derived from the interactions we had in the workshops and generally aimed to simplify the security classification task of our users. We prepared the template in a spreadsheet which contained all the information from the previous ten-steps presentation, albeit in a more structured way.

The spreadsheet template contains the following components:

Step: It shows the step number, which coordinates the attention of the user and helps direct the workflow.

Task: A column providing the task description. The text here is simply adopted from the ten-steps described before.

More details: This column simplifies the task with additional descriptions.

Your Response: In this column, the user would store/provide their input responding to the respective task.

Free Text: For the inputs where users should describe the system or components themselves, they were able to write in their own words.

Dropdown list: For the inputs which were defined in the methodology and required specific item from the list (e.g., connectivity, protection level, presence of security functionality), we provided the dropdown menu for selection. We also applied validation mechanisms so that users are guided to select the valid input.

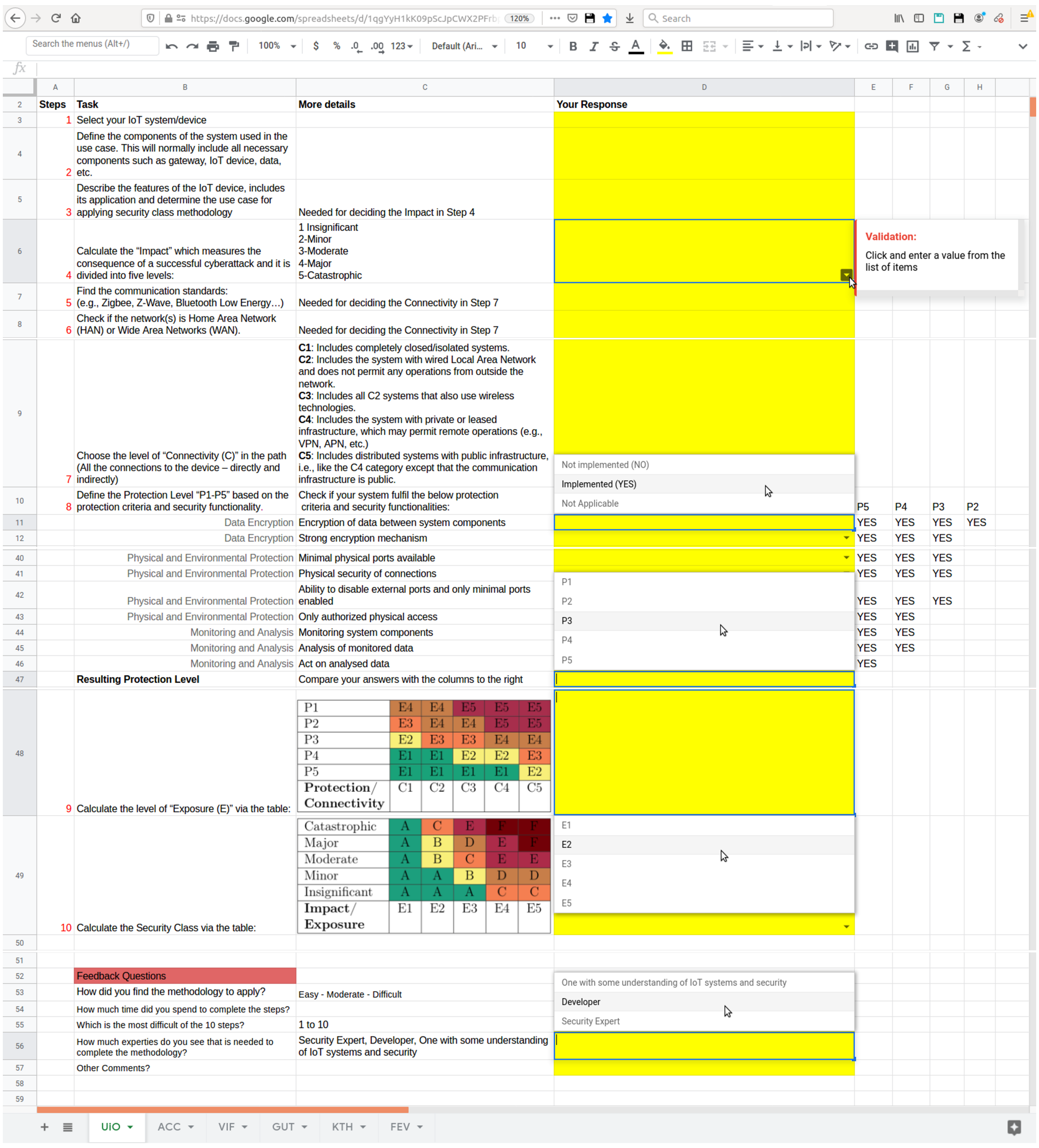
Lookup table: The lookup table was also shown to guide the user to provide valid exposure and security class.

Protection level requirements: There were also columns to show protection level requirements where the users were guided to select the appropriate Protection level (see line 47 in Figure 3).

Spreadsheets can be quite powerful if one knows how to program them. For example, the lookup tables in our last steps could probably be programmed so that users do not need to make the lookup herself, and probably the answers to the long step 8 (note that in Figure 3, several spreadsheet rows have been omitted, i.e., from 13 to 39) could be automatically matched against the protection levels that are listed in the columns to the right. However, we intended the spreadsheet implementation only as a low fidelity prototype, expecting other future versions to follow.

The goal with this first implementation was mostly to have a way to present the ten-steps process to more test users. We planned a project-wide webinar, where the participants that have been helping us with the two examples mentioned before were the main supporters. Therefore, for the spreadsheet implementation, we focused on adding user-friendly aspects to the ten-steps, based on the interactions we had before on the manual paper-based process, mainly focusing on providing clarifying text and the necessary helper information for each step.

The spreadsheet implementation went through one more round of internal testing during a workshop with AVL, one of the SCOTT partners. The result is the one presented in Figure 3 and is the one that we have used to do our final webinar, presented below.



	A	B	C	D	E	F	G	H
2	**Steps**	**Task**	**More details**	**Your Response**				
3	1	Select your IoT system/device						
4	2	Define the components of the system used in the use case. This will normally include all necessary components such as gateway, IoT device, data, etc.						
5	3	Describe the features of the IoT device, includes its application and determine the use case for applying security class methodology	Needed for deciding the Impact in Step 4					
6	4	Calculate the "Impact" which measures the consequence of a successful cyberattack and it is divided into five levels:	1 Insignificant 2-Minor 3-Moderate 4-Major 5-Catastrophic					
7	5	Find the communication standards: (e.g., Zigbee, Z-Wave, Bluetooth Low Energy…)	Needed for deciding the Connectivity in Step 7					
8	6	Check if the network(s) is Home Area Network (HAN) or Wide Area Networks (WAN).	Needed for deciding the Connectivity in Step 7					
9	7	Choose the level of "Connectivity (C)" in the path (All the connections to the device – directly and indirectly)	**C1**: Includes completely closed/isolated systems. **C2**: Includes the system with wired Local Area Network and does not permit any operations from outside the network. **C3**: Includes all C2 systems that also use wireless technologies. **C4**: Includes the system with private or leased infrastructure, which may permit remote operations (e.g., VPN, APN, etc.) **C5**: Includes distributed systems with public infrastructure, i.e., like the C4 category except that the communication infrastructure is public.					
10	8	Define the Protection Level "P1-P5" based on the protection criteria and security functionality.	Check if your system fulfil the below protection criteria and security functionalities:		P5	P4	P3	P2
11		Data Encryption	Encryption of data between system components		YES	YES	YES	YES
12		Data Encryption	Strong encryption mechanism		YES	YES	YES	
40		Physical and Environmental Protection	Minimal physical ports available		YES	YES	YES	
41		Physical and Environmental Protection	Physical security of connections		YES	YES	YES	
42		Physical and Environmental Protection	Ability to disable external ports and only minimal ports enabled		YES	YES	YES	
43		Physical and Environmental Protection	Only authorized physical access		YES	YES		
44		Monitoring and Analysis	Monitoring system components		YES	YES		
45		Monitoring and Analysis	Analysis of monitored data		YES	YES		
46		Monitoring and Analysis	Act on analysed data		YES			
47		**Resulting Protection Level**	Compare your answers with the columns to the right					
48	9	Calculate the level of "Exposure (E)" via the table:	(table below)					
49	10	Calculate the Security Class via the table:	(table below)					
50								
51								
52		Feedback Questions						
53		How did you find the methodology to apply?	Easy - Moderate - Difficult					
54		How much time did you spend to complete the steps?						
55		Which is the most difficult of the 10 steps?	1 to 10					
56		How much experties do you see that is needed to complete the methodology?	Security Expert, Developer, One with some understanding of IoT systems and security					
57		Other Comments?						
58								
59								

Validation:
Click and enter a value from the list of items

P1	E4	E4	E5	E5	E5
P2	E3	E4	E4	E5	E5
P3	E2	E3	E3	E4	E4
P4	E1	E1	E2	E2	E3
P5	E1	E1	E1	E1	E2
Protection/ Connectivity	C1	C2	C3	C4	C5

Catastrophic	A	C	E	F	F
Major	A	B	D	E	F
Moderate	A	B	C	E	E
Minor	A	A	B	D	D
Insignificant	A	A	A	C	C
Impact/ Exposure	E1	E2	E3	E4	E5

Figure 3: Snapshot of spreadsheet implementation of the SCM ten-steps process.

]

5.1.1 Evaluation

We organised a webinar for the whole SCOTT project partners. We used classical methods of advertising to attract many participants, like preparing a text, presenting the webinar (e.g., similar to how one would do for an academic event but more flashy) and emailing an invitation to everyone in the project with reminders, etc. We also worked with the previous partners to make the invitation text to be interesting for our audience, i.e., many of the SCOTT partners were companies.

The plan for the webinar was:

1. An introductory presentation from us, the organisers, which included:
   (a) motivations of the SC methodology, similar to what the research papers were doing,
   (b) a short presentation of the ten-steps process,
   (c) a final exemplification of how we applied the ten-steps to one of the previously mentioned applications (which was from the same project, thus more motivating for the participants).
2. A hands-on interaction from the participants with the online spreadsheet.
3. A brief questionnaire at the end of the spreadsheet (see bottom of Figure 3).

Part 1 took ca. 30min whereas parts 2 and 3 some extra 30-40min, including final discussions.

We had ca. 15 participants in the online webinar (3 were the organisers). The participants were divided into five teams and took the hands-on exercise. Each team had to duplicate the main example sheet (see bottom of Figure 3) and fill it in according to their IoT system of choice. The exercise took between 7-30min to complete. The participants included security experts, developers and system managers having a general understanding of the IoT system and security. We, the organisers, were observing how the teams progress and were answering questions, usually for clarification or acknowledgement.

We also went to the students, using one hour of their exercise classes to do a very similar activity as above, i.e., we presented the spreadsheet tool and asked them to do the same exercise using this tool instead, under our supervision. The ten-steps were now considerably easier to use than in the previous session when only the research papers were given to the students.

5.1.2 Outcomes and Major Observations

From our observations and interactions during the webinar, we could draw the following conclusions.

User help/manual: When users were performing the assessment, even after the spreadsheet and terminologies were explained in our presentation, all users had questions either for clarifying individual steps or assigning values for impact and connectivity.

Automation: Several of the steps could be automated, e.g., determining the protection level, exposure, or class. These were asked for by participants and supported by everyone.

Lack of customisation: The spreadsheet was too static, and it did not allow to change the lookup tables (which participants observed as a necessity when changing the type of system).

Scalability: Spreadsheets are not scalable, both in terms of systems evaluated and in terms of private user space, i.e., allowing users to log in and other classical functionalities that modern tools have.

From the answers to our short questionnaire, we obtained the following:

Moderately difficult: All teams answered that they found the application of the methodology of *moderate* difficulty.

The difficult steps were the evaluation of *impact* and the *protection level.*

Diversity of expertise: Especially for answering all the questions for the protection level, the teams needed diversity of expertise, i.e., they had to ask people that knew about the respective security functionality.

Thus, the major observations were that users needed better help during the assessment, especially for handling the more difficult steps, i.e., for impact and protection levels. Moreover, the next tool should do better in automation, customisation, and scalability, which are usual requirements (and not difficult to obtain) for a web-based application, like the one we present in the rest of the report.

5.2 Introducing the web application

To improve the user experience, we decided to implement the tool as a web application. We did not choose the desktop application to avoid the users' burden of downloading, installing and updating the application in case of changes. It was also easy for us to push new changes without needing the end-user to do special actions. It was also clear that we required data persistence so that the users can perform assessments and save them for future use.

The major technologies used to implement the web-based tool are described below:

ASP .NET Core MVC is a widely used framework for building web applications using MVC (Model View Controller) pattern.[24] To select the technology, our main focus was the speed at which we could produce the prototype. We chose the MVC application because it was quick to start developing and publishing application with clear separation of Model View and the Controller. Moreover, the development platform Visual Studio already provides ready to use templates to create such a web application. We also have implemented a separate service layer so that, if we require to make a public API, we could easily construct a RESTful API to make the tool available to any clients. It would be necessary for integration in a DevOps tool-chain.

We used ASP.NET Core Identity to manage authentication in the application.[25] For responsive user interfaces, we used the Bootstrap framework.

Microsoft Azure is a cloud computing service from Microsoft.[26] It is much easier to manage, configure and deploy web applications using such services. There are other similar solutions from Google and Amazon we well. We selected Microsoft Azure to manage the resources and deploy this SCTv1.

SQL: We used the Azure SQL database for data persistence. In the beginning, we did not require a high-performance database and thus, selected the database with the standard configuration from the Azure portal.

[24]https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-3.1

[25]https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.1&tabs=visual-studio

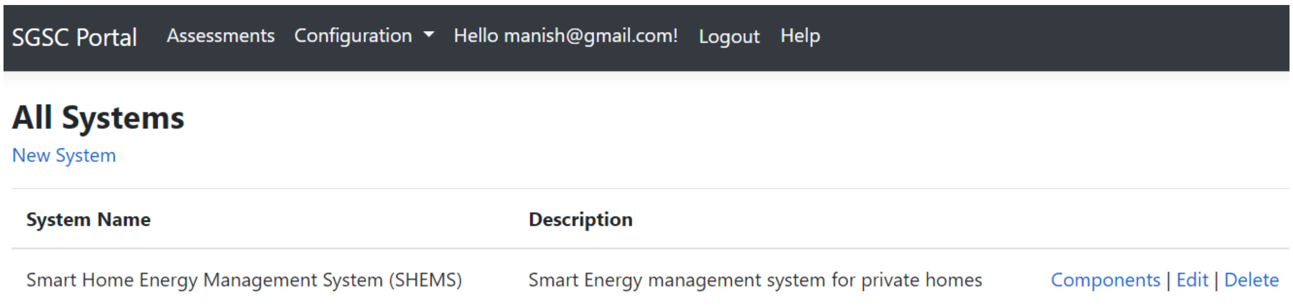
[26]https://azure.microsoft.com/

The tool is hosted as an Azure App service at *https://lightsc.azurewebsites.net/*.

During this work, we simplified the assessment process by combining several of the previous steps into one. Now the core activities that the users must do in the web tool are:

1. **Define a System** (corresponding to step 1 from Section 4.3)

   The user defines the system under evaluation, for which to compute security class. Here the user describes the details of the system and a unique name (to save this evaluation). The details may include how the system works, which technology it uses and what components exist in the system.



All Systems

New System

System Name	Description	
Smart Home Energy Management System (SHEMS)	Smart Energy management system for private homes	Components | Edit | Delete

Figure 4: Snapshot of systems page of SCT web application.

2. **Add components** (corresponding to steps 2–7 from Section 4.3)

   A system is decomposed into its components, and for each component, a class should be computed. A component is described, including information about the role of the component, vendor, communication standards used, etc. One may also include communication capabilities and scenarios where the component is used and how it interacts with other system components. Components should be categorised, where we had as default component types: IoT device, Hub, and Backend System. The user can define their own component types. The component types are relevant for the next step so that the tool can select some of the security functionalities automatically as 'not applicable'.

   The user is also required to define the connectivity level of the component.

   Similarly, the impact of security breaches should be specified here.

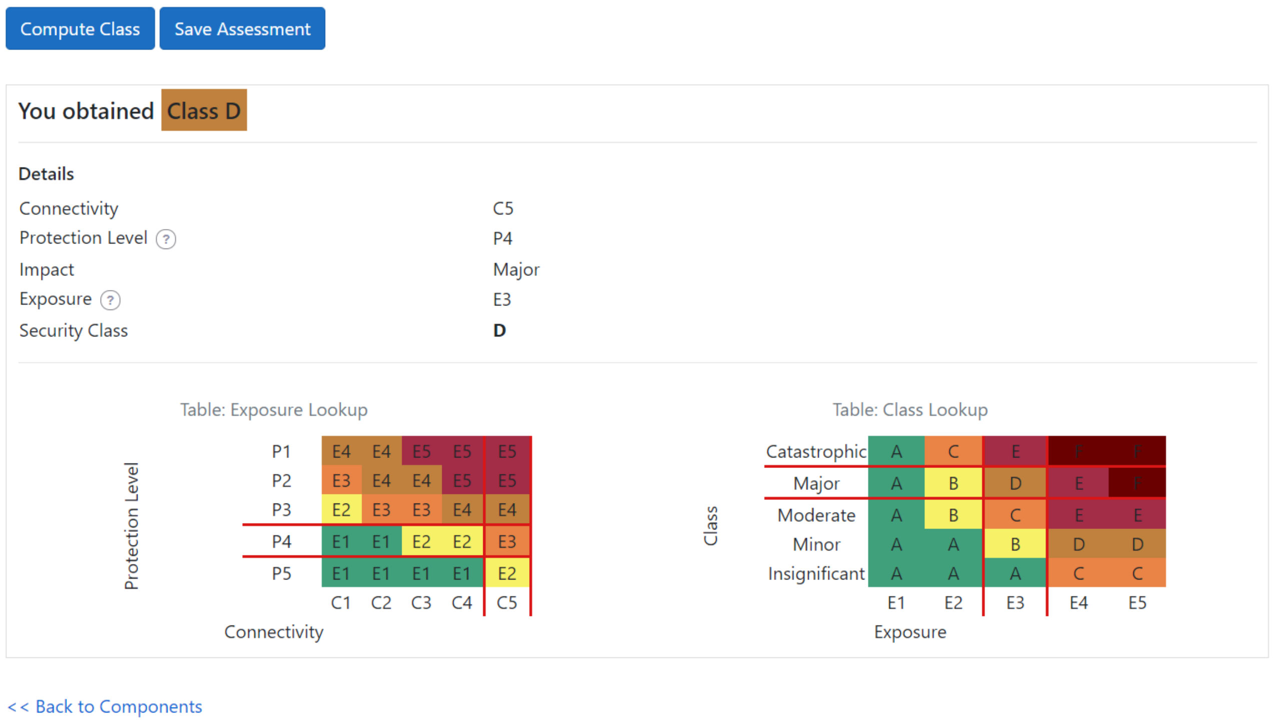
3. **Perform assessment** (corresponding to step 8 from Section 4.3)

   Here the user selects the security functionalities present in the system, which are required for determining the protection level.

4. **Compute class** (corresponding to steps 9–10 from Section 4.3)

   After all the inputs are provided for a component, the user can compute the security class by pressing a button (see Figure 5). The selected parameters for the security functionalities are used to compute the protection level and then using the lookup tables to calculate exposure level further and ultimately, the security class for the given component. Figure 5 shows the final view containing the lookup tables and selections made to obtain the resulting class.

Besides these tasks, the user can also save the assessment for future reference. Using the tool, the user should be able to browse the assessment and perform CRUD operations on them.



You obtained Class D

Details

Connectivity	C5
Protection Level	P4
Impact	Major
Exposure	E3
Security Class	**D**

Table: Exposure Lookup

Protection Level	C1	C2	C3	C4	C5
P1	E4	E4	E5	E5	E5
P2	E3	E4	E4	E5	E5
P3	E2	E3	E3	E4	E4
P4	E1	E1	E2	E2	E3
P5	E1	E1	E1	E1	E2

Connectivity

Table: Class Lookup

Class	E1	E2	E3	E4	E5
Catastrophic	A	C	E	F	F
Major	A	B	D	E	F
Moderate	A	B	C	E	E
Minor	A	A	B	D	D
Insignificant	A	A	A	C	C

Exposure

Figure 5: Snapshot of class calculation view.

5.2.1 Evaluation and Observations

The application was demonstrated both to the students and the SCOTT partners AVL and GUT. For the students, we again presented in a lecture and demonstrated how the web application could be used, and they took it as an exercise to use the tool on their IoT systems from the course. The SCOTT partners were two of our main interaction users, whom we used throughout this work. We had one workshop where we presented (similarly as we did for the students) the new web application and demonstrated how to apply it to the original SHEMS example from research paper [20] (which has always been our first example for each of our implementations and tests). After the presentation, and during the demonstration, we had a long period of discussions with comments from the users.

The improvements have been appreciated, especially the save functionality and the login possibility since it allowed for a private space for someone to work with their evaluations. The automation was as expected.

The negative comments were especially related to the lack of help and guidance. One specific request was to have tool-tips for various parts of the interface so to tell them what was that button/text was about.

5.3 Second version of SCT

The final version of the web application had the following extra usability functionalities:

1. **Customisable lookup tables.** Lookup tables are usually constructed by experts. The default ones that the application offers are the ones we took from the research papers of the SCM [22, 20].

   However, depending on the domain of application, the lookup table may differ slightly. Therefore, one should be able to change the lookup table according to the domain. The tool has a configuration feature where the user could override the default lookup table and also reset it to default.

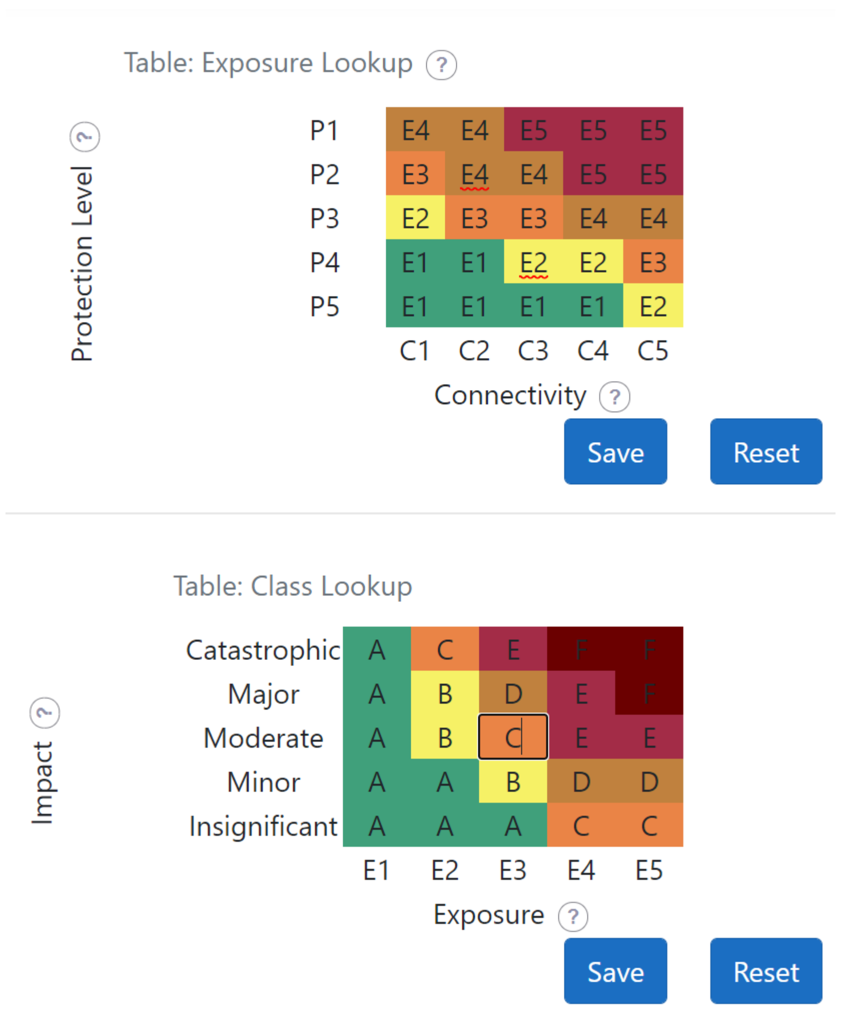


Table: Exposure Lookup

Protection Level	C1	C2	C3	C4	C5
P1	E4	E4	E5	E5	E5
P2	E3	E4	E4	E5	E5
P3	E2	E3	E3	E4	E4
P4	E1	E1	E2	E2	E3
P5	E1	E1	E1	E1	E2

Connectivity

Table: Class Lookup

Impact	E1	E2	E3	E4	E5
Catastrophic	A	C	E	F	F
Major	A	B	D	E	F
Moderate	A	B	C	E	E
Minor	A	A	B	D	D
Insignificant	A	A	A	C	C

Exposure

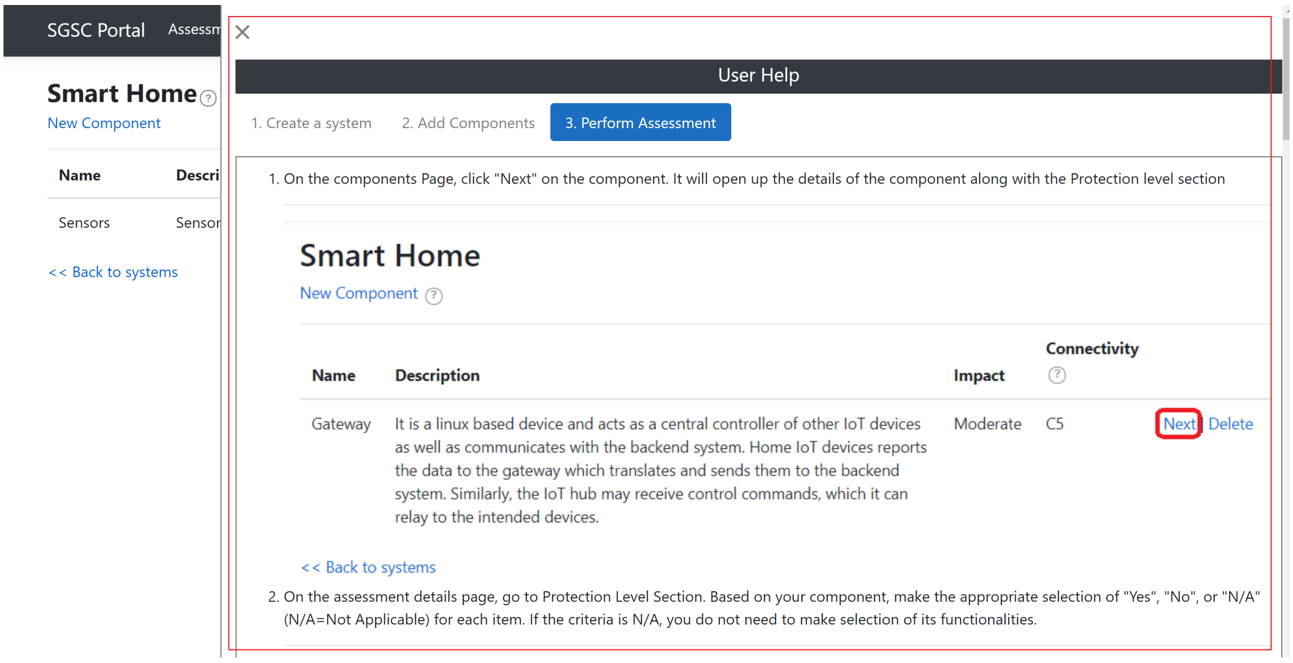
Figure 6: Snapshot showing the customisation of lookup tables.

2. **Main user guide easily available on every page.** The preliminary tool had a user guide only on the landing page. Every time the user needed help, they had to browse to that page, which was considered hectic.

   The final tool has a help sidebar menu which on click, slides over the page the user help (see Figure 7). This sidebar allows the user to focus on their tasks, without the distraction of opening a new page each time help is needed.

3. **Detailed contextual help.** The test users demanded detailed explanations of the terminologies and the steps. We added help icons beside the text that required detailed descriptions of the terminology or step. Clicking on the help icon a modal opens up to show these details. Many of these details also appear in the main help. Figure 8 shows the modal for describing the connectivity types.

To be able to perform the assessment, the user first needed to create an account. At this point, we also decided to implement the *weights* aspect of the SC methodology from the research paper [21]. Before, in the spreadsheet, it was challenging to work with weights; but now we could more easily calculate using weights. The tool was doing the calculations, implementing the formulas from [21]; whereas the user had only to specify the individual weights.


SGSC Portal Assessm

Smart Home
New Component

Name	Descri
Sensors	Sensor

<< Back to systems

User Help

1. Create a system 2. Add Components 3. Perform Assessment

1. On the components Page, click "Next" on the component. It will open up the details of the component along with the Protection level section

Smart Home
New Component

Name	Description	Impact	Connectivity	
Gateway	It is a linux based device and acts as a central controller of other IoT devices as well as communicates with the backend system. Home IoT devices reports the data to the gateway which translates and sends them to the backend system. Similarly, the IoT hub may receive control commands, which it can relay to the intended devices.	Moderate	C5	Next Delete

<< Back to systems

2. On the assessment details page, go to Protection Level Section. Based on your component, make the appropriate selection of "Yes", "No", or "N/A" (N/A=Not Applicable) for each item. If the criteria is N/A, you do not need to make selection of its functionalities.

Figure 7: Snapshot showing user help opened in a sidebar from the right.

5.3.1 Evaluation through a Hackathon

Helped by the SCOTT partner GUT (Gdansk University of Technology) we organised a hackathon contest with a cluster of Polish SMEs. The cluster, we were told by our Gdansk contact person, had in the order of 100 technology companies.

The *preparations* for the hackathon included:

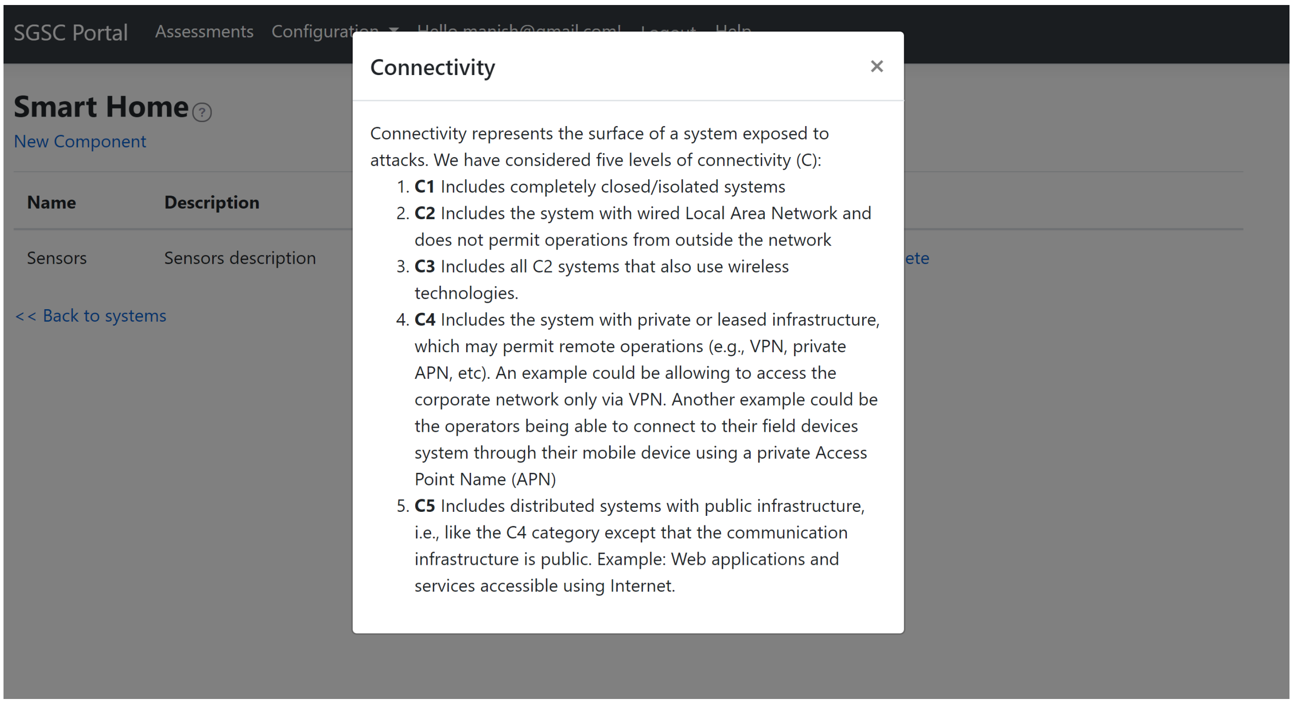
1. preparing a video tutorial (ca. 10min) on how to use the tool;
2. preparing a presentation with slides
   (a) motivating the concept of security classification,
   (b) describing the benefits for industry,
   (c) explaining the ten-steps process, and
   (d) how to apply it to the SCOTT pilot (this we mostly reused from previous workshops with additions and adaptations to fit the target audience);
3. materials for announcing and attracting participants and for managing the contest.

The hackathon day had a ca. one hour program, which was recorded through the online meeting tool with:

1. a short introduction (2min) from the SCOTT official and the Polish cluster official (our contact point),
2. followed by our presentation and demonstration of the web tool,
3. ending with the presentation of the contest, rules, tasks, and prizes (described further).

The hackathon format included a contest with three prizes (winning 2000€ in total) and rules for participation and evaluation. Our purpose with preparing such a complicated setup was firstly to attract diversity in the participating teams, as well as hoping to increase the number of participants.

The *contest* asked the teams to



Connectivity

Connectivity represents the surface of a system exposed to attacks. We have considered five levels of connectivity (C):

1. **C1** Includes completely closed/isolated systems
2. **C2** Includes the system with wired Local Area Network and does not permit operations from outside the network
3. **C3** Includes all C2 systems that also use wireless technologies.
4. **C4** Includes the system with private or leased infrastructure, which may permit remote operations (e.g., VPN, private APN, etc). An example could be allowing to access the corporate network only via VPN. Another example could be the operators being able to connect to their field devices system through their mobile device using a private Access Point Name (APN)
5. **C5** Includes distributed systems with public infrastructure, i.e., like the C4 category except that the communication infrastructure is public. Example: Web applications and services accessible using Internet.

Figure 8: Snapshot showing help text for connectivity levels in a modal component.

1. use the tool on one of their systems or components;
2. describe how the security classes could contribute to innovation and business potential for their company; and
3. devise an innovative way of using the SCM within their companies' technical, business, or management processes.

The winner would be *evaluated* based on a report where the above should be described, putting weight on innovation.

Besides the above contest format, the hackathon also included a *usability part*.

1. We offered special recognition prizes (with cash winnings too) for those that take substantial effort to help us with the usability studies, i.e., to use the two aspects mentioned below. These prizes were separate from the contest prizes and were advertised as optional.
2. We prepared a survey and asked the participants to take part in the survey, which was available through a special menu in the web interface. Figure 9 shows a screenshot of the survey as appearing to a user. We have omitted some questions to fit it within a single page. The survey included questions regarding user experience, opinion about the tool, facts about the users, their expertise and knowledge of DevOps, and further suggestions.
3. We used a tool called Hotjar[27] to track and analyse the activities during the evaluation. Hotjar offers several features to perform the usability evaluation of a website. However, our focus was only to see how users work with our tool. Thus, we found the following strategies relevant to evaluate our tool. See details for all of these in Section 5.3.3 on page 24.

   - **Screen recordings.** Recording the activity of the user while working with the web tool was captured anonymously, for concerns of privacy and consent. Because of this,

[27]https://www.hotjar.com/

were not able to correlate the recording to the survey. There were more recordings than people who took the survey.

- **Incoming feedback.** Using this feature, the users could select the specific part of the page and provide feedback on it.
- **Heatmaps.** Heatmaps show which part of the page was clicked, scrolled or moved the most. Using this feature, we might be able to identify which features the users are most interested or are most difficult and require most effort/time.

The participation was unexpectedly poor: We had only four companies attending the hackathon one-hour presentation and only three that submitted a document for evaluation. Moreover, only one participated in the survey. Because of this, we continued to test with individual users selected personally, as detailed below in Section 5.3.2. After the hackathon day, we released the recording of the meeting (containing our presentation and contest description) as well as the tutorial video. The goal was that the local contact person would replay this, and send the information out again, during an official cluster meeting that was scheduled to come soon. Still, no more participants were registered. The explanation that we later received from the local contact person was described as *"Language barrier"*, i.e., The writing in English was discouraging, and the internationalisation that the hackathon offered was not of interest since many of the cluster companies had already a large client base in Poland.

5.3.2 Evaluation with Individuals

Since we aimed to evaluate how easy it was to use the tool in the system development life cycle, we also asked feedback from software professionals (the last group of users described in Section 3). We selected technically sound individuals and experts in product development, but not necessarily in security. In particular, we wanted individuals with different roles such as CEO, CTO, consultant, architect, or system developer. We prepared a list of probable participants and reached out to them through emails. The individuals were mostly employees from eSmart Systems AS and Smart Cognition AS, both of which are software companies. We tried to organise the workshop to introduce the tool to them. However, it was not possible because of their availability. However, for two of the participants, we were able to describe the tool in person, in two separate meetings. Thus, we sent out emails with the necessary materials to perform the assessment and asked them to contact us if they need help with understanding the concept. The following materials were provided:

1. URL to access the tool;
2. Presentation slide to understand the core concept of SCM and SCT (reused from the hackathon);
3. URL to access the video tutorial presenting how to use the SCT;
4. URL to user help and terminologies;
5. Description of the task that we were asking them to perform to complete an assessment. We described that the evaluation is complete after the test users, at minimum, create a system, add sub-systems to this system, perform the SC assessment of each sub-system to calculate the class, and finally take the survey. We also asked them to provide feedback (incoming feedback in Hotjar) while using the tool if they had any. They were free to browse any other part of the application on their own interest.

Five individuals took part in the evaluation. Other uninterested participants responded by saying that they have no inputs to this tool as it is not their field, some said that they were busy, and some did not respond to emails at all.

5.3.3 Outcomes and Major Observations

Following results were obtained from Hotjar:

Heatmap: The heatmap of the assessment page showed that the main help menu was clicked only 0.1% of the time. However, the user help available on each component was clicked frequently. Similarly, another most clicked part of this page was the compute class button (5.6%). It shows that users were interested in computing the class quite often, most probably because they were repeating short cycles of changing some parameters and recompute the class. Figure 10 shows an example of a heatmap for the assessment page. One of the least components that users interacted with was the belief and weight inputs in the assessment page. Though the help icon to explain their concept was fairly clicked, the input box was rarely clicked.

Screen recordings: Screen recordings showed that the majority of users used the tool as expected. They first create the account and browse through the description and then check the main help page. After that, they follow the instruction of creating the system and adding sub-systems. Most of the users follow a similar pattern of browsing the pages and clicking on the help icons to see the details and understand better what to select. It also showed that most users did not interact with the belief functionalities (and thus left these be the default ones). Some looked into the help icon of belief and weights, but most of them did not change any values of beliefs during the assessment.

Incoming Feedback: We expected many users to take part in providing such local feedback since these seem familiar from social media sites like Twitter or Instagram, which people like. In our experience, providing feedback through the Hotjar functionality is relatively easy: the user just needed to click on the feedback button on the right side of the page, provide the smiley rating, write a comment, and click the send button. If they wanted, they could select the HTML page component about which they wanted to provide the feedback. Surprisingly, only one user provided incoming feedback. Figure 11 shows an example of incoming feedback.

Survey: The survey showed that the users were entirely new to the classification methodology and took 30 to 100 minutes to apply it. However, some users did not keep track of the time that they used to complete the assessment. Similarly, learning this tool, the maximum amount of time used was 15 to 60 minutes. One of the users who had some security background only used 3 minutes to learn it. It was probably because of the familiarity with security terminology, and also he had an individual workshop session with us, where we gave a presentation and a demonstration of the tool.

Similarly, the tool was considered usable in the planning phase of the product by most users. Figure 12 shows the survey result of the question "In which of the DevOps phases do you think this security classification tool (or parts of it) can be used?".

Most of the participants found the belief evaluations to be the most unintelligible part of the tool. The same result was confirmed by the heatmap evaluations and the user recordings as they were the least interacted components on the page. Surprisingly, three of the five users found the system definition section, where one defines the system and sub-systems, difficult.

Three of the users considered that with a basic understanding of security, one could apply this method. Similarly, one of them considered that software developers could apply this methodology. However, one said that it requires the skill of security experts to apply this methodology.

Out of five users, four found the methodology moderately easy. However, one of the users found it difficult to apply in his system because the user considered that the individual security assessments are not easy without deeper knowledge of the concepts that are being evaluated. However, he considered that the methodology was easy to understand. Similarly, all the users considered it easy to find the help that they needed while using the application. One of the constructive feedbacks was to provide more guidance to fill in the confidence parameters.

6 Final evaluation results and major observations

The last observations from the Hotjar tracking and survey, including both the one response from the hackathon and the five from the individuals, are generally suggesting that the main part of the SC tool is easy enough to be used by non-security experts. The more experimental weights and beliefs part of the tool was considered not so easy and is regarded as a complex feature also in the research papers for the SCM.

In the hackathon contest, we have received three submissions, each applying our SC tool to one IoT system and describing the innovations that the SC methodology and tool can bring to their companies. Also, these could be evaluated to understand more the use of our tool; we have disregarded the other aspects that the hackathon contest asked for, i.e., innovation and business aspects. One application was to a Mini Unmanned Surface Vessel, and they used the SCT to compare between a not secured version, that resulted in class F, and a secured version which resulted in class B, which helped them understand what security functionalities the system needed. The other two application done during the hackathon were to analyse the security of autonomous vehicle management systems in logistics and to RFID. Both of these reports similar uses of the tool, i.e., for trying out different security features for different configurations of their systems resulting in different security classes.

In total, throughout all our stages of creating the tool, we saw the SCM applied to many IoT systems, which we could count as: The last version of the SCT was used by three SCOTT partners on their respective systems, VEMCO, PHILIPS, and VTT. The hackathon saw the SCT used on three systems. The individual 'Software experts' used the SCT on five systems. The students used (multiple version of) the SCT on two systems. The SCOTT partners during the webinar using the spreadsheet low-fidelity prototype have applied the SCM to five more systems. We, the organisers, have applied different versions of the SCT for exemplification purposes in different stages to one system. This totals to ca. 19 applications of the SC methodology, done mostly by non-security experts or teams, through the use of our different prototype implementations. These provided valuable feedback regarding the usability of the original method and of the prototypes that we have been building. The final prototype has been considered fairly easy to work with.

The principles of a DevOps-ready Security Classification from Section 2.2 have motivated our work. We have implemented the chosen methodology into a tool, thus respecting the Principle 2, and we have worked and tested to make this tool easy to use for non-security experts (i.e., our choice of users was as such), thus respecting Principle 3. We did not strive in the direction of Principle 1. However, having now a tool, one can at least do manual re-evaluations of the system by making the necessary changes in the evaluation. We have made the tool so that it can also provide an API, but did not work in that direction as this is an engineering task that is best left to a software development company to undertake. However, we believe that Principle 1 can easily be attained once having a tool as the one that we have demonstrated and tested. We leave this as further work, to be done by companies willing to take our SC tool, or similar ones, into their DevSecOps tool-chains, since the adjustments and implementations are routine.

A general recipe has emerged, we believe, for going from a research effort security classification to a DevOps tool. Any such endeavour, inspired by the present work, would include three main phases:

1. Make a step-based process out of the published security classification methodology.
2. Test it in a low-fidelity computer-based implementation, where we have seen that the spreadsheets are very good for this purpose.
3. Implement the high-fidelity tool, like the web-based version that we did, where more of the process is hidden behind a natural interaction process with the tool that guides the user to the final class.

This is something very familiar to the interaction design field, but not so familiar to the security tools developers and researchers. Choosing well the target group representatives, and testing these three minimal passes is essential.

7 Conclusions and Related works

We this report, we identify five principles for a security classification methodology to be DevOps-ready, i.e., ready to be used in a DevSecOps tool-chain. Debatable as they might be, these principles are viewed as initial guidelines. The major part of this work is concerned with exemplifying the existing security classification methodology to satisfy the five principles. To do this, we create a tool that implements the chosen methodology, thus conforming to Principle 2. We also tested its usability showing how it conforms with Principle 3. We have detailed our process of evaluating such a tool for its usability, which involved participants from industry applying the various tool prototypes at different stages to ca. 19 IoT systems, during ca. 14 workshops and larger events, involving as test users, both teams and individuals for ca. 9 months.

From the process that we have detailed in both Section 4 (for the manual work with the methodology) and Section 5 (for the tool prototypes), we could extract a general recipe detailed in Section 6. This simple guide can be applied to other 'tool-ification' endeavours done for similar security methodologies. We particularly encourage such activities since we see an increased need for usable security tools and methods, demanded by the DevSecOps culture, which is becoming popular in software development companies.

The tool in itself is a contribution, as it expands the user group from security experts to non-experts, and it reduces the time used for designing and evaluating the security of IoT systems. Using such tools, companies can utilise existing internal resources (i.e., their developers or CTOs) to evaluate the security of their system. It is not only that more people can contribute to making the IoT products more secure, but also more people can now use a security tool to understand what it means for a product to be secured and how to achieve that.

7.1 Related Work

7.1.1 NOR-STA

NOR-STA (also called trust cases) is an argumentation tool to support compliance, assurance and security cases [7]. During the assessment, the assessor can assign the Confidence and

Decision parameters to build confidence. Confidence and decision are the qualitative scales where confidence shows the degree of confidence on the statement, and the decision shows the degree to which the statement is acceptable. These scales are later aggregated to provide overall confidence and decision of the whole assessment. The argumentation model they use is called trust-IT model and is based on Toulmin's argument model. The aggregation of confidence and decision parameters are based on Dempster-Shafer theory. The tool is sophisticated and has many features that we wanted for security classification method (e.g., argumentation model, confidence parameters, aggregation mechanisms etc.). However, we could not use it to implement security classification methods for the following reasons:

1. **Limited to compliance against strict predefined requirements**

   In order to use the tool, the security requirements, along with detailed security functionalities, should be clearly defined before the assessment. After that, one can argue whether the security requirement is fulfilled or not. Thus, it can only be used against compliance requirements which are strictly defined beforehand. However, the security classification methodology is more flexible and cannot be restricted to the security functionalities before the assessment. Of course, we can set the goal class in the beginning. However, there are multiple ways to reach the given class by controlling impacts, connectivity and protection mechanism parameters. Thus, to be able to use this tool, one must define each alternative as templates and then have to assess each of the templates to find where the system under evaluation fits. Thus, NOR-STA is not practical for security classification method where one can reach the same level of security class using several alternatives.

2. **Scaling function**

   NOR-STA specifies confidence parameters based on Dempster-Shafer theory. Thus, it assigns belief and plausibility parameters for each argument and aggregates them to obtain overall belief and plausibility. However, users cannot assign this scale themselves. Instead, they use so-called VAA approach where the user assigns the confidence using linguistic decision and confidence scale, which is then converted into belief and plausibility for aggregation. The aggregated belief and plausibility is converted back to aggregated confidence and decision scale. To perform these conversions, they use a scaling function which they claim was obtained from the experimental evaluation. However, they do not specify what value to use for parameters for conversion, and thus the tool acts as a black box, and the results cannot be replicated without using the tool. Since the scaling function results were not transparent, the values for parameters to calculate scaling function may depend on the domain of assessment or entirely on experts. Thus, NOR-STA did not fit our needs.

3. **Aggregation of confidence parameters**

   NOR-STA specifies different aggregation rules for beliefs. In some rules, it considers the weighted mean approach for Complimentary arguments. However, the weighted mean approach is not sensitive to extreme lower values. Similarly, in SC and NSC arguments, the aggregated result is obtained by multiplying individual beliefs. That means the resulting belief will always diminish even though more pieces of is added to support the claim. Thus, we propose to use the Multi-Metrics approach for aggregation, which NOR-STA does not support.

7.1.2 Certware and other Eclipse plugins

Certware is an open-source eclipse plugin-based tool from NASA [3]. It supports the development of safety, assurance and dependability cases [28][29]. One of our interests were the creation of argumentation model using GSN diagrams. However, Certware is also not feasible for implementing security classification because they also fit the strict requirements to perform compliance. For flexible requirements such as in security classification, it requires generating several templates which is not feasible. However, the generation of diagrams from the assessment is useful and can be one of the parts of the tool-chain.

Some examples of similar open-source tools for building cases using GSN diagrams are D-case editor [30] and Acedit [31]. No changes have been found in the source code of these tool in recent years.

References

[1] Christopher Alberts, Audrey Dorofee, James Stevens, and Carol Woody. Introduction to the OCTAVE Approach. Technical report, Carnegie-Mellon University, Software Engineering Institute, 2003.

[2] Ross Anderson and Shailendra Fuloria. Certification and evaluation: A security economics perspective. In *2009 IEEE Conference on Emerging Technologies & Factory Automation*, pages 1–7. IEEE, 2009.

[3] Matthew R Barry. Certware: A workbench for safety case production and analysis. In *2011 Aerospace conference*, pages 1–10. IEEE, 2011.

[4] Barry W Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, 1988.

[5] Irena Bojanova and Jeffrey Voas. Trusting the internet of things. *IT Professional*, 19(5):16–19, 2017.

[6] Alistair Cockburn. *Agile software development: the cooperative game.* Pearson Education, 2006.

[7] Lukasz Cyra and Janusz Gorski. Support for argument structures review and assessment. *Reliability Eng. & System Safety*, 96(1):26–37, 2011.

[8] Jennifer Davis and Ryn Daniels. *Effective DevOps: building a culture of collaboration, affinity, and tooling at scale.* O'Reilly, 2016.

[9] Viktor Farcic. *The DevOps 2.0 Toolkit.* Packt Publishing Ltd, 2016.

[10] Virginia NL Franqueira, Zornitza Bakalova, Thein Than Tun, and Maya Daneva. Towards agile security risk management in re and beyond. In *Workshop on Empirical Requirements Engineering (EmpiRE 2011)*, pages 33–36. IEEE, 2011.

[28] https://nasa.github.io/CertWare/
[29] https://github.com/nasa/CertWare
[30] https://github.com/d-case/d-case_editor/
[31] https://github.com/arapost/acedit/

[11] Rune Fredriksen, Monica Kristiansen, Bjørn Axel Gran, Ketil Stølen, Tom Arthur Opperud, and Theodosis Dimitrakos. The CORAS framework for a model-based risk management process. In Stuart Anderson, Massimo Felici, and Sandro Bologna, editors, *International Conference on Computer Safety, Reliability, and Security*, pages 94–105. Springer, 2002.

[12] Jez Humble and David Farley. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.

[13] Jack Jones. Factor analysis of information risk, August 6 2004. US Patent App. 10/912,863.

[14] Minhaj Ahmad Khan and Khaled Salah. IoT security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 82:395–411, 2018.

[15] Gene Kim, Jez Humble, Patrick Debois, and John Willis. *The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution, 2016.

[16] UL LLC. Methodology for marketing claim verification: Security capabilities verified to level bronze/silver/gold/platinum/diamond. Technical report, UL LLC, 2019.

[17] Y. Lu and L. D. Xu. Internet of Things (IoT) Cybersecurity Research: A Review of Current Research Topics. *IEEE Internet of Things Journal*, 6(2):2103–2115, 2019.

[18] Steve McConnell. *Rapid development: taming wild software schedules*. Pearson Education, 1996.

[19] Jason RC Nurse, Sadie Creese, and David De Roure. Security risk assessment in internet of things systems. *IT professional*, 19(5):20–26, 2017.

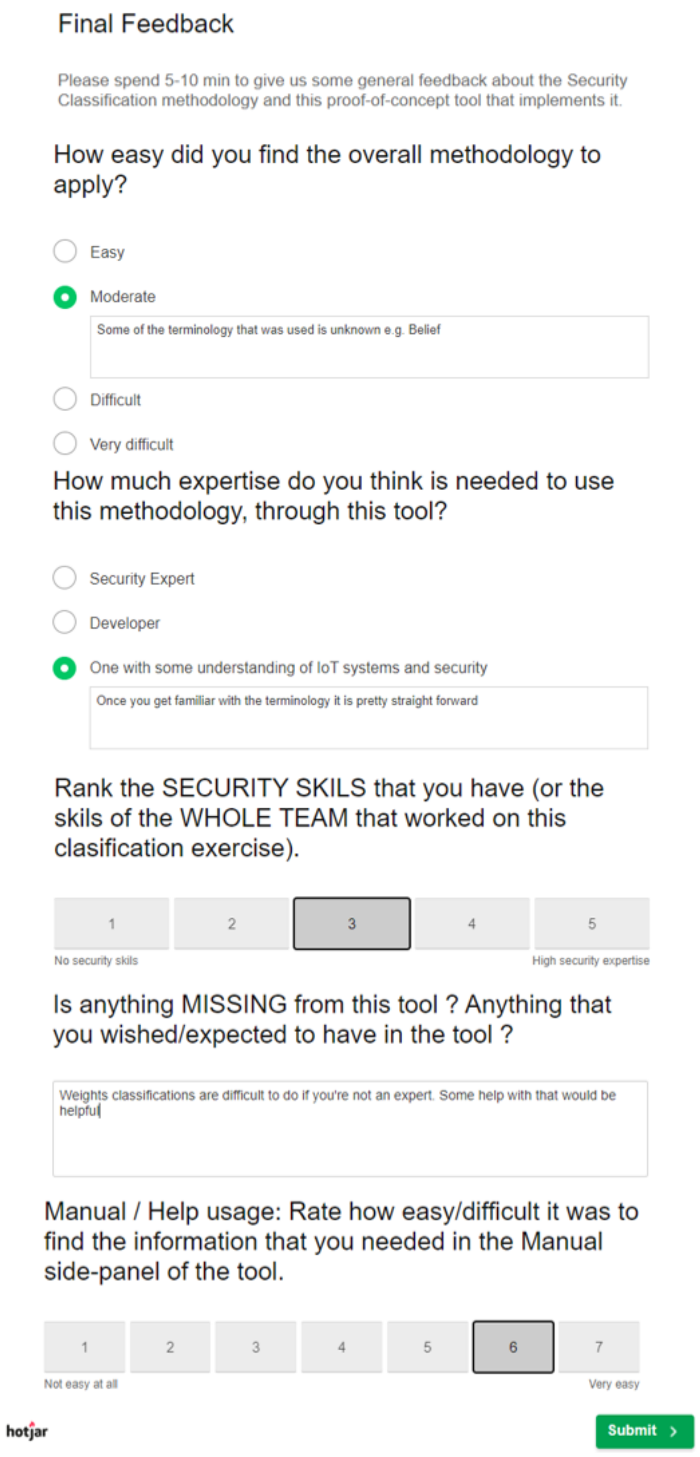
[20] Manish Shrestha, Christian Johansen, and Josef Noll. Criteria for Security Classification of Smart Home Energy Management Systems. In *Int. Conf. Smart Information & Comm. Technologies*. Springer, 2019.

[21] Manish Shrestha, Christian Johansen, and Josef Noll. Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT. In *5th International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 244–249. IEEE, 2020.

[22] Manish Shrestha, Christian Johansen, Josef Noll, and Davide Roverso. A Methodology for Security Classification applied to Smart Grid Infrastructures. *International Journal of Critical Infrastructure Protection*, 28:100342, 2020.

[23] Stefan Taubenberger, Jan Jürjens, Yijun Yu, and Bashar Nuseibeh. Problem analysis of traditional it-security risk assessment methods–an experience report from the insurance and auditing domain. In *IFIP International Information Security Conference*, pages 259–270. Springer, 2011.

[24] Zhi-Kai Zhang, Michael Cheng Yi Cho, and Shiuhpyng Shieh. Emerging Security Threats and Countermeasures in IoT. In *10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, page 1–6. ACM, 2015.



Final Feedback

Please spend 5-10 min to give us some general feedback about the Security Classification methodology and this proof-of-concept tool that implements it.

How easy did you find the overall methodology to apply?

Easy [✓ Moderate] Difficult Very difficult

Some of the terminology that was used is unknown e.g. Belief

How much expertise do you think is needed to use this methodology, through this tool?

Security Expert Developer [✓ One with some understanding of IoT systems and security]

Once you get familiar with the terminology it is pretty straight forward

Rank the SECURITY SKILS that you have (or the skils of the WHOLE TEAM that worked on this clasification exercise).

1 2 [✓ 3] 4 5

No security skils — High security expertise

Is anything MISSING from this tool ? Anything that you wished/expected to have in the tool ?

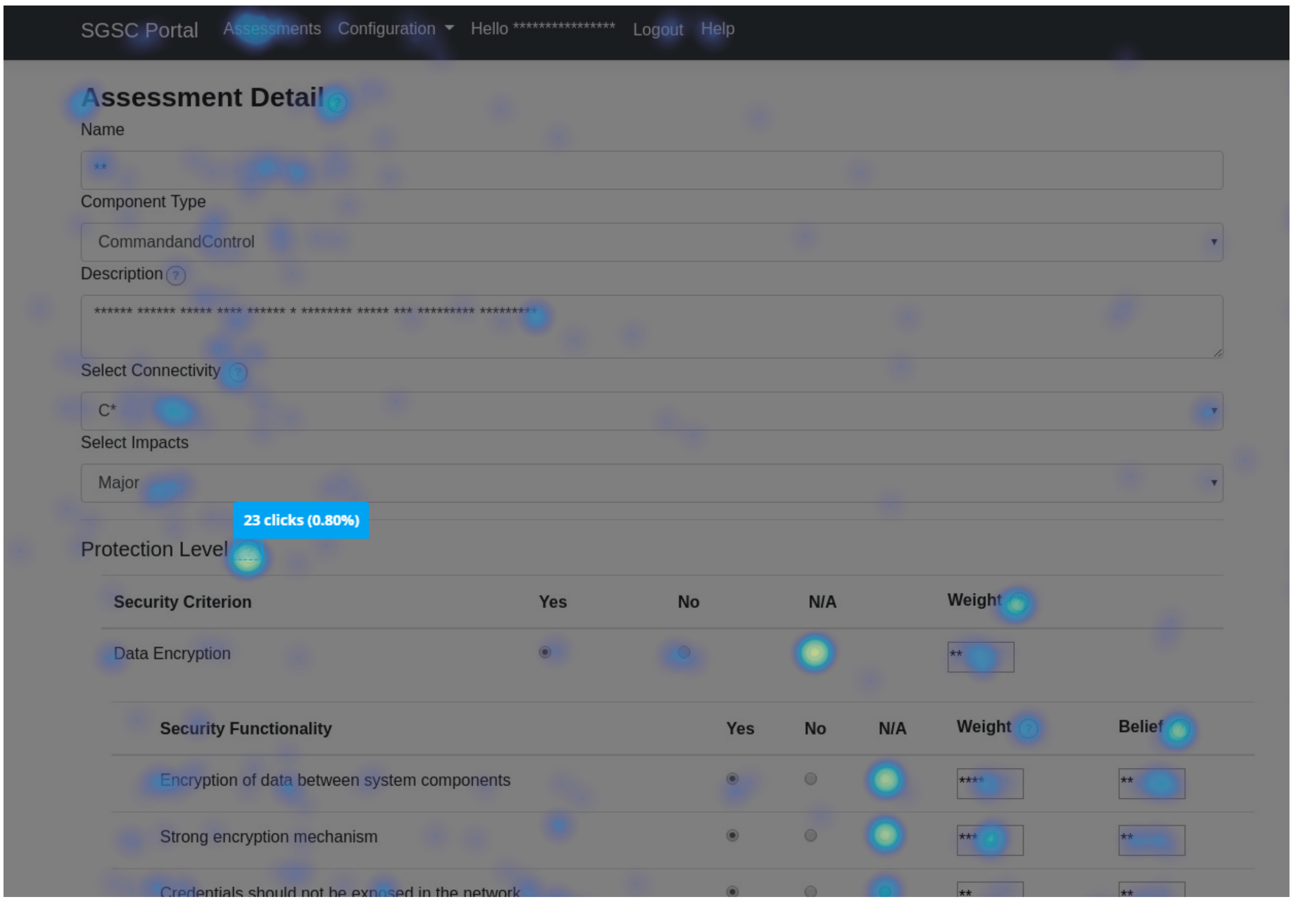
Weights classifications are difficult to do if you're not an expert. Some help with that would be helpful

Manual / Help usage: Rate how easy/difficult it was to find the information that you needed in the Manual side-panel of the tool.

1 2 3 4 5 [✓ 6] 7

Not easy at all — Very easy

Figure 9: Snapshot of five of the questions from the survey.



Assessment Detail

Name

**

Component Type

CommandandControl

Description

****** ****** ***** **** ****** * ******** ***** *** ********* ********

Select Connectivity

C*

Select Impacts

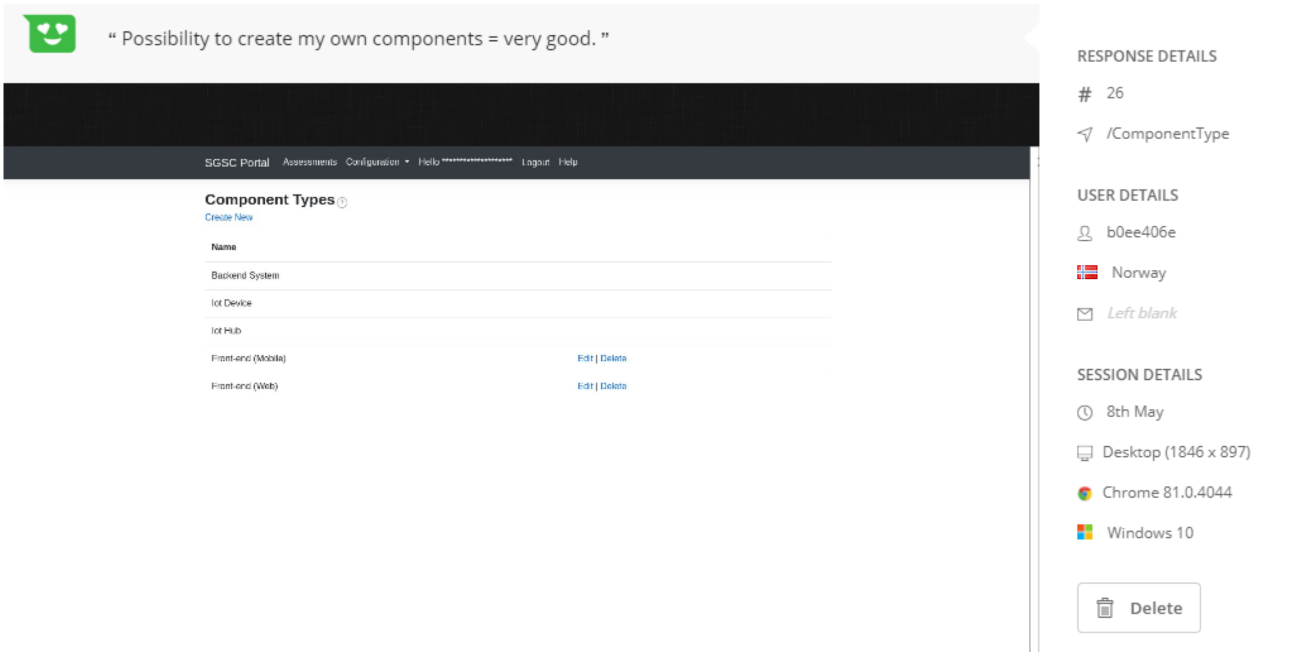
Major

Protection Level

Security Criterion	Yes	No	N/A	Weight
Data Encryption	●	○	○	**

Security Functionality	Yes	No	N/A	Weight	Belief
Encryption of data between system components	●	○	○	****	**
Strong encryption mechanism	●	○	○	***	**
Credentials should not be exposed in the network	●	○	○	**	**

Figure 10: Snapshot taken on the admin-side of Hotjar, showing a heatmap.



“ Possibility to create my own components = very good. ”

Component Types

Name	
Backend System	
Iot Device	
Iot Hub	
Front-end (Mobile)	Edit | Delete
Front-end (Web)	Edit | Delete

RESPONSE DETAILS

26

/ComponentType

USER DETAILS

b0ee406e

Norway

Left blank

SESSION DETAILS

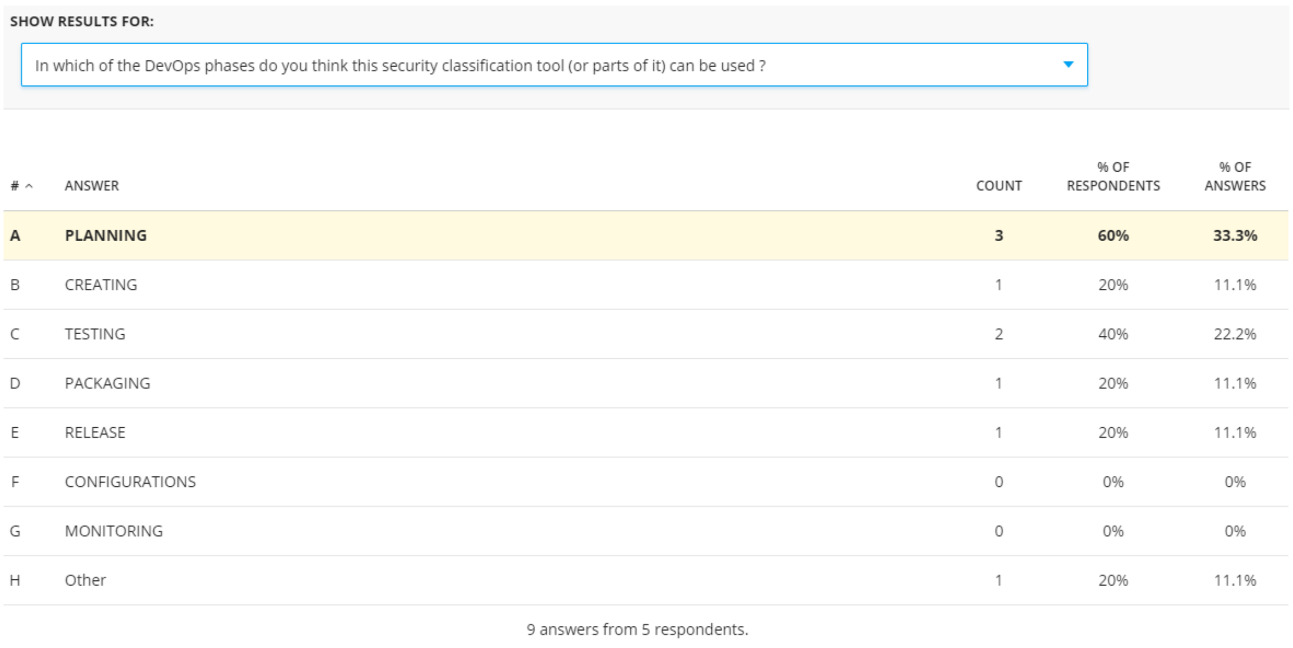
8th May

Desktop (1846 x 897)

Chrome 81.0.4044

Windows 10

Figure 11: Snapshot taken on the admin-side of Hotjar, showing an incoming feedback.



SHOW RESULTS FOR:

In which of the DevOps phases do you think this security classification tool (or parts of it) can be used ?

#	ANSWER	COUNT	% OF RESPONDENTS	% OF ANSWERS
A	**PLANNING**	**3**	**60%**	**33.3%**
B	CREATING	1	20%	11.1%
C	TESTING	2	40%	22.2%
D	PACKAGING	1	20%	11.1%
E	RELEASE	1	20%	11.1%
F	CONFIGURATIONS	0	0%	0%
G	MONITORING	0	0%	0%
H	Other	1	20%	11.1%

9 answers from 5 respondents.

Figure 12: Survey answer on the usability of the tool in different phases.