UiO : **University of Oslo**

Manish Shrestha

LightSC: A light-weight security classification methodology to design and evaluate security of IoT systems

Thesis submitted for the degree of Philosophiae Doctor

Department of Technology Systems
Faculty of Mathematics and Natural Sciences

University of Oslo

2021

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 1234

ISSN 1234-5678

Cover: Hanne Baadsgaard Utigard.
Print production: Reprosentralen, University of Oslo.

To Rozina and Reon

Abstract

The Internet of Things (IoT) is ubiquitous and is an essential part of human life. We are constantly surrounded by an IoT enabled environment and use their services, at home, at work, and during travel. In addition to the value of services IoT systems provide, the exposure of these systems creates security and privacy challenges. IoT systems were designed to fulfill functional purposes and with little concern for security. Its overwhelming exploitation in diverse domains has raised security and privacy challenges. Several stakeholders have come together to focus on securing IoT systems, which has resulted in building new communication protocols with security in focus. However, many IoT systems continue to be insecure, and security incidents continue to increase. We have observed that IoT systems in practice lack adequate security implementation despite available mechanisms. Current security approaches are cumbersome and require substantial involvement of security experts, which makes the security process expensive for low-cost consumer IoT systems. Also, based on the criticality of the IoT systems, the security requirement may differ. For instance, the security requirement of a pacemaker is much higher than that of a temperature sensor. Thus, this thesis aims to construct a security evaluation tool, allowing for an easy assessment of security given as a security goal in mind. The approach is designed to be straightforward, easy, and applicable even by non-security-experts, which reduces the dependency of system designers on the security experts. In particular, this research proposes a light-weight methodology called LightSC, which is tailored towards non-security-experts, who can make decisions in selecting appropriate connectivity and security mechanisms for a desired level of security. We have validated the methodology by analyzing real IoT systems such as Advanced Metering Infrastructures (AMI) and Smart Home Energy Management Systems (SHEMS), using the LightSC methodology. Moreover, the thesis also points out several application areas that can use the LightSC methodology and open up the methodology in new fields, such as integration of automation in LightSC, and application of LightSC in DevSecOps cycle. We also have implemented tool support to make the LightSC methodology DevSecOps ready, which was evaluated by real stakeholders (mostly non-experts) by using the LightSC tool successfully to evaluate nineteen different IoT systems. The valuable feedback regarding the usability of the LightSC tool can be seen as useful proof of the applicability of the original security classification methodology that we have proposed.

Preface

This thesis is submitted to the Department of Technology Systems, Faculty of Mathematics and Natural Sciences, University of Oslo, in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* at the University of Oslo. The research presented here was conducted at the University of Oslo and eSmart Systems AS. My main supervisor has been Christian Johansen, Associate Professor at the University of Oslo. Professor Josef Noll at the University of Oslo and Chief Analytics Officer Davide Roverso at eSmart Systems AS has been my co-supervisors. This work is supported by eSmart Systems AS and the Research Council of Norway through the projects IoTSec – "Security in 1420 IoT for Smart Grids", with number 248113/O70, and MeasurEGrid – "Measurable Security and Privacy for Services on the Smart Electricity Grid", with number 259635.

The thesis is a collection of four papers, presented both in academic conferences and submitted to journals, and technical reports that specify the development process from the design of the methodology up to the implementation. The papers are preceded by introductory chapters that relate them together and provide background information and motivation for the work.

Acknowledgements

First and foremost, I am grateful to the Department of Information Technology Systems at the University of Oslo (formerly UNIK), Norwegian Research Council, and eSmart Systems AS for supporting my PhD project through funding and facilities. Without them, my PhD journey would not have begun.

I want to express the deepest gratitude to my supervisor, Dr. Christian Johansen. I am grateful for his precious time and guidance through each stage of my PhD. Without his persistent guidance and belief in me, the PhD journey would not have taken this form. I want to express my sincere thanks to my co-supervisor, Professor Josef Noll, for his constructive feedback and scientific discussions and, most importantly, advice on the problems through different perspectives. I am grateful to Dr. Davide Roverso, my co-supervisor at eSmart Systems AS, for his encouragement and advice from the very beginning of my PhD project. I am very grateful to Professor Olaf Owe for his help, wisdom, and encouragement. His calmness and humbleness will always be an inspiration to me. I would also like to thank the SCOTT project (WP21) for their collaboration.

I would like to appreciate the valuable conversation and exchange of thoughts with Michael Pacevicius, which provided me with a lot of energy to remain persistent. I thank Ehale Fazeldehkordi, Dr. Dang Ha The Hien and, Dr. Nhan Van Nguyen for all the fun and memories shared during the PhD journey. I am grateful to Heidi Bjerke for her kindness and helpfulness.

I want to thank my wife, Rozina Dongol, for her emotional support and patience, which kept me going. Last but not least, I am grateful to my parents for their love, trust, and encouragement.

Manish Shrestha
Oslo, March 2021

List of Publications

Papers

Paper I

Shrestha, M., Johansen, C., Noll, J., and Roverso, D. (2020). A methodology for Security Classification applied to Smart Grid Infrastructures. International Journal of Critical Infrastructure Protection, 28, 100342. DOI:10.1016/j.ijcip.2020.100342

Paper II

Shrestha, M., Johansen, C., and Noll, J. (2019, September). Criteria for Security Classification of Smart Home Energy Management Systems. In International conference on smart Information & communication Technologies (pp. 157-165). Springer, Cham. DOI: 10.1007/978-3-030-53187-4_19

Paper III

Shrestha, M., Johansen, C., and Noll, J. (2020, April). Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT. In 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC) (pp. 244-249). IEEE. DOI: 10.1109/FMEC49853.2020.9144957

Paper IV

Shrestha, M., Johansen, C., Moghadam, M. D., Johansen, J., and Noll, J. Security Classification for DevSecOps: Five Principles and an Exemplification. (Submitted to ACM Transactions on Privacy and Security (TOPS)).

Technical Reports

Technical Report I

Shrestha, M., Johansen, C., and Noll, J. (2017). Security Classification for Smart Grid Infrastructures (long version). Research report http://urn.nb.no/URN:NBN:no-63576.

Technical Report II

Shrestha, M., Johansen, C., and Noll, J. (2019). Criteria for Security Classification of Smart Home Energy Management Systems (long version). Research report http://urn.nb.no/URN:NBN:no-79526.

Technical Report III

Shrestha, M., Johansen, C., and Noll, J. (2020). Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT (long version). Research report http://urn.nb.no/URN:NBN:no-80474.

Technical Report IV

Shrestha, M., Johansen, C., Moghadam, M. D., Johansen, J., and Noll, J. (2020). Tool Support for Security Classification for Internet of Things (long version). Research report http://urn.nb.no/URN:NBN:no-84145.

Contents

Part I

Overview

Chapter 1

Introduction

1.1 Motivation

The Internet of Things (IoT) is the network of physical devices that embed sensors, electronics, software, and network connectivity enabling the devices to collect and exchange data about their operation and environment, and be remotely controlled. IoT, as a field of study, addresses the move towards a sensor-driven infrastructure for automated processes where standardized interfaces connect cheap sensors with networks, service platforms, and applications. The evolution of IoT has created transformative opportunities. Smart grids, health care, assisted living, and public safety are a few of the many sectors to be dominated by IoT platforms.

IoT systems are designed and manufactured at a low cost and have limited memory and processing power, making them attractive to cyber attackers. Despite the security approaches available today, the number of attacks is increasing, and many IoT systems are still insecure. For instance, in 2013, Fouladi and Ghanoun showed implementation problems in Z-Wave [25]. During their analysis of a Z-Wave door lock, they found that all 16 bytes of the temporary key used to exchange the encryption key were all zeros, allowing them to decrypt the encryption key [25]. Similarly, Zillner and Strobl pointed out that for Zigbee, even though it is a standard that has robust security features, the main challenge comes from the poor and minimal implementation of security features [62]. The same claim is still supported in the recent work of Celebucki et al. [11], where they compared the security of Zigbee, Z-wave, and Bluetooth and revealed that despite all security protocols having built-in security options, the security features are not implemented in practice. It clearly shows that, although the security in communication protocols is considered relatively good, the security features are either not implemented or poorly implemented.

We consider that cost may be one of the significant factors for failing to prioritize security in IoT systems. There is also a lack of guidance or a structured approach to building secure systems. The security analysis performed by Trujano et al. revealed that the DJI Phantom 3 Standard drones had the hard-coded root password and the default Wi-Fi password, which shows that despite several security features implemented in the drone, several baseline security recommendations are ignored [58]. In the Phantom 4 Pro version, most of the vulnerabilities in Phantom 3 were fixed. However, they still had vulnerabilities such as GPS spoofing [15]. While new threats and vulnerabilities are expected to emerge, designing systems with a security mindset can help overcome avoidable flaws beforehand (e.g., Phantom 3 issues).

There are a few recently emerging standards for security in IoT systems, most notably ETSI TS 103 645 and ISO/IEC 27030. Standards are known to be expensive and time-consuming, involving extensive documentation and significant efforts from experts to be established. There are also certification and labeling schemes for IoT products typically based on standards and are verified via penetration and vulnerability tests. Thus, security standards and certification programs are expensive and usually take at least a year to obtain. Though IT systems may benefit from such approaches, IoT devices being designed and produced as low-cost systems, with a shorter life cycle, investing in such programs is not suitable for IoT.

Expensive certification programs may benefit critical systems where an extreme level of security assurance is required. However, security requirements vary with the criticality of the system. Thus, a one-size-fits-all approach toward security certification is not favorable. Instead, we need flexibility to choose a desired level of security. There is a need for a framework that can guide the selection of security features based on the domain needs, rather than selecting and applying security features without any specific goal and claiming that the security is reasonable.

Guidelines are only seen as recommendations and not necessarily proof of how secure a product is built. Furthermore, fulfilling all recommendations from the best practices may not be practical either. Processes such as risk analysis are also cumbersome and require significant involvement of security experts, and should be carried out regularly, which is often seen to be expensive for consumer IoT products.

To overcome the barriers mentioned above, we first need to simplify the security approach so that non-security-experts can contribute to security assessment. Secondly, we should be able to specify the security level indicating the applicability of the system with respect to security goals under consideration. This measurable approach not only motivates the selection of just enough protection for a targeted system but also raises awareness among the stakeholders to choose the correct vendor that fits their context. This encourages to establish a shared responsibility among all participants to secure a system.

1.2 Methods used in the thesis

The work presented in this thesis has been guided by the standard computer science research method, where the presentation of Rober L. Glass [31] is especially relevant. This work is more aligned with the engineering method, which includes the following steps:

1. observe existing solutions to a problem,
2. propose better/improved solutions,
3. build or develop (a prototype/artifact),
4. measure and analyze (to test and validate the newly proposed solution),

5. repeat until no further improvements are possible.

The last step (repeat until no further improvements are possible) is rarely followed [31], we also are not inline with that step and rather repeat until "satisfying" results are obtained. More specifically, the research methodology can be presented as follows:

1. In the first stage, we aimed to have a broader understanding of the IoT security domain, having a particular focus area on Smart Grids and Smart Home Energy Management Systems (as part of the Smart Grid). The choice of application field was also motivated by the close collaboration with the IoTSec national Norwegian project and the active involvement and interest of the company eSmart Systems AS, the problem owner in this project. We carefully observed various approaches and current practices in IoT security (Smart Grid security in particular) to precisely identify the challenges that industries face in terms of security. Therefore, we tried to attack a broader problem. Deeper investigations involved reading relevant literature and discussions with the domain experts in the industries about the problems they have been facing.

2. We eventually narrowed down the problem and set the research goals that are detailed in Section 1.3. In short, we wanted to develop a method to allow any IoT systems to be developed secure-by-design, which was especially useful for such a fast-moving field. The particular focus on Smart Grids, which is a critical infrastructure, attracted our attention towards security classification methods. However, good as these may be for critical systems, the classification methods are not ideal for security-by-design, and especially so for cheap IoT systems in an agile market.

3. To address the research goals, we have proposed and developed an improved solution, which consisted of an enhanced security classification methodology, extending the well-known ANSSI classification method towards IoT systems and security-by-design. Particularly, we have departed from the attack-centric and traditional risk assessment style of the ANSSI classification. Instead, we introduced the idea of a *functionality-centric* approach, which we consider to be better suited for developers to embrace security-by-design in an agile software development team to develop IoT systems.

4. The solution, which in the beginning was called SGSC (for Smart Grid Security Classification – being a methodology only), was then further analyzed and evaluated, conducting also surveys and case studies.

5. The results of our validation efforts have been used to enhance our initial solution, going through several iterations, each adding different new contributions/improvements to the original methodology. Eventually, this process culminated with the creation of a proof-of-concept prototype implementation of the methodology in the online tool that we have called LightSC. This prototype has been further evaluated for usability since one

of the main claims of our methodology (and hence of this tool supporting it) was that it was “easy-to-use by non-security-experts”. The final validation results are encouraging, as detailed in the rest of this section.

Besides the classical peer-reviewing process that was used to check our results detailed in the published papers, we have also used the industry partners from the project IoTSec (but also reaching out to a larger European project called SCOTT, for more IoT interested industries) to test our ideas by presenting and getting feedback from them.

We have built on previous methods, including the published research on measurable security and multi-metrics, as well as on the ANSSI classification proposals. Staying close to existing solutions allows for our proposal to be easily accepted and taken into use. Moreover, reusing and building upon existing (research) works allows moving the field much further.

For the usability evaluations, we have used several of the classical methods from usability studies, including structured interviews, observations, guided workshops, etc. Having such a controlled, method-based process helped ground well our final claims of “easy-to-use” by non-experts for our final LightSC prototype. This grounding allows our claims of usability to be trusted. Therefore, our LightSC can be considered the first security classification method that can be used by software developers in a fast-paced development style as modern DevOps teams use when creating IoT systems. We thus encourage DevOps teams to take up in use our DevOps-ready LightSC tool.

1.3 Research Goals

The primary goal of this research is to develop a light-weight approach to design and evaluate the security of IoT systems against a targeted security goal, bringing up measurable security aspects. The second goal is to demonstrate the applicability of such a method in real systems to ensure that the approach is usable in the IoT system development life cycle. The following research questions guide the thesis to achieve the research goals:

RQ1 How can we ease the creation of secure IoT systems?

RQ2 How can we measure the trustworthiness of the proposed method?

RQ3 How can we enable system engineers and non-security-experts to evaluate the security of their systems and provide them with solutions being compliant with the envisaged security goals?

RQ4 How can a tool simplify the use of the proposed methodology?

To accomplish the above goals, first, we adopt the notion of a *security class*. Unlike most methods being attack-centric, our approach is functionality-centric and system-centric, where one can select the appropriate security mechanisms for their systems or system components to reach the desired security class. The security classification methodology that we propose is built upon the

Agence nationale de la sécurité des systémes d'information (ANSSI) standard methodology for security classification of Industrial Control Systems (ICS). This methodology is related to risk analysis methods with the difference that it has the purpose of assigning a security class to the system based on (combinations of) scores given to the various exposure aspects of the system and the respective protection mechanisms implemented; without looking at attackers.

To show the applicability of our methodology, we selected the Advanced Metering Infrastructures (AMI), which was in the process of being rolled out in Norway, as our first case study. AMI refers to the integrated metering system with communication capabilities and can automatically perform the measurement, collection, delivery, analysis, and storage of metering values. We analyzed the AMI topology to identify the impact and exposure (physical and IT) aspects of the infrastructure and derived the security criteria for secure AMI systems. We used a real example of a smart meter installment with an unprotected physical button to turn on and off the smart meter to argue that protecting the device is not enough; the context of its deployment should also be considered. Thus, the security classification methodology is driven by the philosophy of system-level security, which is achieved by decomposing the system into its sub-systems and components considering their interactions. Each system element is analyzed to offer a better system security design and evaluation of complex IoT systems such as smart grids. We also discussed the usability of our goal-centric and system-centric approach towards regulatory bodies, companies, and end-users.

We further enhanced the security classification methodology for consumer IoT systems by defining security criteria and protection levels. We then demonstrated the applicability of the classification methodology by using it to evaluate and improve the security of an existing Smart Home Energy Management Systems (SHEMS) from class D to class A. However, like most approaches, the evaluation using security classification is also subjective. Hence, to justify the decisions and improve the assurance of the assessment, we further introduced confidence parameters into the methodology using belief and uncertainty. We exemplified the calculation of confidence parameters for a use case involving an edge command and control mechanism for SHEMS.

Finally, we propose five principles for a methodology to be DevOps-ready so that it is applicable to non-security-experts (system designers and developers). To further validate against the identified DevOps principles and to improve the methodology, a tool to support the security classification method was developed. Several stakeholders evaluated the tool by applying it to several existing IoT systems being developed. Successfully applying the methodology to several real systems by the stakeholders showed the validity of our methodology. The evaluation of the tool also provided directions toward improving the methodology itself as well as the user interface.

1.4 Structure of the thesis

The rest of this thesis is structured as follows: Chapter 2 provides the background information required to understand the thesis. It starts with the Internet of Things and discusses the trends towards the security of IoT systems. In particular, we discuss the use of risk assessment and assurance cases in IoT systems. Emerging standards, certifications, and labeling schemes in the IoT domain are also discussed. Chapter 3 describes the scientific contributions of this work. It introduces the security classification methodology and shows its validity by applying the methodology in several real IoT systems by non-security-experts. It also outlines the usability test performed against the tool support developed for the methodology. Chapter 4 presents a summary of the academic publications collected in the second part of the thesis. Chapter 5 concludes this overview part by summarizing the contributions and answering the research questions presented in this chapter. Finally, it gives insights into the directions for further research, where we discuss the implications of our research, especially in the DevSecOps[1] life cycle. Part II includes the research papers building the basis for this PhD research.

[1] https://devsecops.org

Chapter 2

Approaches towards securing IoT systems

This chapter provides the necessary background information relevant to this thesis. It first analyzes why IoT systems have specific security needs, focusing on the weaknesses of current solutions for IoT security. Then, we provide an overview of state of the art in security approaches of IoT systems.

2.1 Security of IoT Systems

IoT has driven technology and computing abilities into the smallest devices. The capability of embedded devices to connect to the Internet to send and receive data has made these more alive and powerful, making data ubiquitous in both the collection and the availability side. According to International Data Corporation (IDC), by 2025, the number of IoT devices would grow to 41.6 billion and generate 79.4 zettabytes of data[1].

The proliferation of IoT-enabled devices has created new and transformative opportunities. IoT is widely adopted in significant sectors, including critical infrastructures such as smart grids and privacy-sensitive domains such as smart homes. Rehman, Asif, and Ahmad described current and future application domains of IoT technology [53]. Similarly, Asghari, Rahmani, and Javadi provided the taxonomy to classify the IoT application domains into health care, environmental, smart city, commercial, industrial, and general aspects [5]. They also found that health care (29%) and smart city (20%) have the highest percentage of application domains.

Data created by IoT-enabled devices at home, at work, or while moving, creates security and privacy concerns. Failing to protect such sensitive data can have adverse impacts. Molina-Markham et al. showed that just by using high-frequency consumption data from the smart meters, one could reveal the privacy-sensitive information such as home occupancy, sleeping & eating routines of the residents [47]. Breach of such information to criminals may result in burglaries or, at worst, life-threatening crimes. Similarly, Greveler et al. revealed that the smart meters under their investigation transmitted unencrypted sensitive data [35]. They also demonstrated privacy infringement by using high-frequency smart meter consumption data to successfully recognize the behavior patterns and even the TV channels watched.

Multiple inter-dependencies, uncertainties, and dynamic interactions give rise to a complex risk picture, especially since applications and networks of IoT systems were initially designed for purely functional purposes (e.g., connectivity,

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

control, and sensing), without cybersecurity and privacy consideration, leading to vulnerabilities that are challenging to address.

IoT devices are typically resource-constrained and consist of heterogeneous networks. Thus, implementing traditional security mechanisms are challenging. Hossain, Fotouhi, and Hasan classified the constraints of implementing secure IoT devices as hardware, software, and network limitations [37]. Further challenges in securing IoT systems comes from the human aspect. The end-users lack security awareness, and many cannot appropriately differentiate between secure and insecure products, and designers also lack the incentives for building secure IoT systems [14]. The survey performed by Blythe, Sombatruang, and Johnson confirms that one of the reasons consumers cannot select an adequately secure product is the lack of sufficient and interpretable information from the vendors about the security of the IoT product to make the security-informed purchase decision [9]. The authors also demand to summarize the security of IoT products through customer-friendly mechanisms such as introducing labeling schemes.

Standardization, certification, and risk management are emerging approaches towards securing IoT systems. Despite efforts to ensure security, reports show that attacks in IoT systems are increasing[2,3]. Best practices and compliance frameworks are also emerging. However, existing security management approaches are expensive and time-consuming, and require a substantial amount of documentation that requires a significant amount of time from experts (high dependency on experts). Investing in such methods does not pay off because of the lower cost and short life span of IoT products.

Thus, the baseline of our work is that securing IoT systems is only envisaged when (i) end-users demand security compliance, and (ii) system engineers have solutions available, allowing them to create secure IoT systems without being security experts. The next section will discuss the major existing approaches towards the security of IoT systems, also motivating the need for our novel approach.

2.2 Risk Assessment

Risk assessment is a planned process followed to find possible breaches into a system, consider the relevant ones, and devise a plan to fix them. Categorization of risks and how to handle them is typically done via specialized risk assessment methodologies or frameworks, most of them typically being based on ISO 31000 and ISO/IEC 27005 standards [52]. ISO/IEC 27005 is based on ISO 31000, but it provides guidelines specifically for Information Systems. Examples of risk

[2] https://blog-assets.f-secure.com/wp-content/uploads/2019/09/12093807/2019_attack_landscape_report.pdf

[3] https://symantec-enterprise-blogs.security.com/blogs/expert-perspectives/istr-2019-internet-things-cyber-attacks-grow-more-diverse?om_ext_cid=biz_social3_AMS_NAM-IV_twitter_

assessment frameworks include CORAS [27], EBIOS[4] from ANSSI, TVRA[5] from ETSI, FAIR [38], and OCTAVE [1].

Traditional risk-assessment methods follow the waterfall model and cannot compete in the IoT era, where technology and threats change rapidly. Taubenberger et al. analyzed the traditional IT-security risk assessment methods and claimed that traditional risk assessment methods are biased, inflexible, static, and highly manual [56]. These issues are valid for IoT systems as well, and thus, we need a better approach towards IoT systems. They also claim that we need security requirement-based assessments and assurance as the foundation, which the present research takes into account.

The decisions in risk assessment are primarily based on the expert's experience and knowledge. Therefore, there is no way to verify whether all threats and vulnerabilities have been correctly identified. A historical event-based likelihood also does not work for the new systems as there is no indication of threats coming up in the future, which may be completely different.

An attack-centric philosophy drives risk assessment. However, attacks are changing too often in IoT. Therefore, the current risk analysis methods may not keep up with the changing threat landscape [49]. Besides, the decisions are based on how likely an attack might occur. Nevertheless, for the IoT world, if the device is exposed, it is very likely to be attacked. Thus, though security testing and risk analysis are useful to identify threats and vulnerabilities, the system should be securely designed in the first place, following the security by design principle.

2.3 Standards and Certifications

Standards are the documents or guidelines for doing something, typically developed jointly by the stakeholders, including experts from the industries, regulators, and consumers who benefit from it. Bojanova and Voas point out several challenges of IoT systems [10], where the lack of standards and certifications and a loose effort from the regulatory side are considered major challenges. To address these challenges, Voas and Laplante recommend the following three IoT certification areas: product, people, and process [59]. They indicate that certifying IoT system components only, do not consider the interaction between system components and thus demands the system certification to obtain a higher level of trust. They also claim that IoT security certification is essential but challenging to achieve with adequate confidence.

There are also a few labeling schemes, such as BSI Kitemark and Underwriter Laboratories (UL) security rating, that raise the awareness and motivation of end-users to secure their system. Moreover, it also motivates the manufacturers

[4]ENISA, "EBIOS". https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/rm-ra-methods/m_ebios.html.

[5]ETSI Technical Specification *"ETSI TS 102 165-1: Method and proforma for Threat, Vulnerability, Risk Analysis (TVRA)"*, 2017. https://www.etsi.org/deliver/etsi_ts/102100_102199/10216501/05.02.03_60/ts_10216501v050203p.pdf

to produce secure systems with compliance according to the security requirement defined by the regulatory bodies.

2.3.1 ISO/IEC 27000 series

ISO/IEC 27000 series includes several standards typically related to the Information Security Management Systems (ISMS) published jointly by the International Organization for Standards (ISO) and Internal Electrotechnical Commission (IEC). The standard ISO/IEC 27030, which is currently under development, specifically deals with the security and privacy for IoT by providing the guidelines towards baseline security requirements. Other ISO/IEC standards, such as ISO/IEC 27001 and ISO/IEC 27002, are more related to the management of overall cybersecurity of an organization and specifies a series of activities towards managing the information security risk.

2.3.2 Common Criteria

The Common Criteria (CC) is an international standard for computing security certification. It was first developed as a standard security evaluation framework to avoid re-evaluating products in the global market. Thus, the evaluation standards from the USA (the Orange Book), Canada (CTCPEC), and European countries (ITSEC) merged to form the common criteria standard, which was first issued in 1994[6]. Common Criteria has also been accepted as an international standard ISO/IEC 15408.

Certifications, like the Common Criteria, are expensive [39], even more so for IoT [7, Sec.III], and takes a long time to obtain, e.g., [41, p.4] report the need for 45.9 person-years to achieve EAL7 for their micro-kernel of 8700 lines of C code.

Common Criteria offers an assurance continuity mechanism which includes re-evaluation and maintenance activities to handle changes in the systems [7]. Though this approach takes less time and costs less than the initial evaluation, these may still be a significant factor, e.g., frequent patches or enhancements with notable changes may lead to the invalidation of the certification of the new version of the product and therefore being difficult to maintain over time [3].

2.3.3 Security Assurance Levels

SIL or Safety Integrity Level is the number assigned to the safety function of a given system [51]. It has four different levels, the highest number representing the highest reliability of safety functions. Similar to the concept of SIL, the International Society of Automation (ISA99) introduced Security Levels (SLs), which later changed to Security Assurance Levels (SALs) [30]. SALs were introduced for securing industrial automation and control systems, and provide a qualitative scale for security. Like SIL, it has four assurance levels (SAL1-SAL4). Each level represents the increasing security features targeting different threat

[6] https://www.commoncriteriaportal.org/iccc/ICCC_arc/history.htm

[7] https://www.commoncriteriaportal.org/files/operatingprocedures/2012-06-01.pdf

actors and is based on the strength of attackers. For instance, SAL1 aims to protect against causal or coincidental violation, whereas SAL4 is targeted towards protecting against intentional violation using sophisticated means with extended resources [30]. The evaluation of SAL is based on the seven foundational requirements: access control, use control, data integrity, data confidentiality, restrict data flow, timely response to an event, and resource availability [30].

2.3.4 ETSI TS 103 645

European Telecommunications Standards Institute (ETSI) is an organization responsible for the standardization of Information and Communication Technologies (ICT) in Europe. Standards from ETSI are recognized internationally and not limited only to Europe. In 2019, ETSI published the Technical Specification (TS 103 645) for consumer IoT entitled "Cyber Security for Consumer Internet of Things" [22]. Its specifications are based on the guidelines specified by the UK Government's "Code of Practice for Consumer IoT Security" [13]. The TS 103 645 provides guidelines to the stakeholders involved in the development and manufacturing of consumer IoT for securing their IoT systems. This specification provides the best practices for security using thirteen high-level provisions for securing consumer IoT products.

The TS 103 645 is further utilized to produce the European Standard "CYBER; Cyber Security for Consumer Internet of Things: Baseline Requirements" or EN 303 645 [23], which was approved in April 2020.

2.3.5 ENISA guidelines

European Union Agency for Cybersecurity (ENISA)is the European agency that contributes to improving cybersecurity by providing advice and solutions to EU member states and stakeholders. It also helps in the development and implementation of EU cybersecurity policies. ENISA has been involved in establishing several guidelines and baseline recommendations for IoT and smart grids. It defines the appropriate security measures for smart grids in [16], which suggests ten different domains similar to ISO 27002 objectives (with the possibility to add new domains). Each domain needs to be labeled with sophistication levels between one and three, representing how much of the security goals of a given domain have been achieved. However, the three sophistication levels are too narrow, and they specify the rating of domains, not the system or the system components. Similarly, in its report [17], it provides the security recommendation for IoT systems, and in [18], it discusses the best practices of the software development life cycle for IoT systems. These guidelines from ENISA are useful in defining requirements; however, prioritizing and designing systems based on the criticality of the systems are yet to be defined. It also focuses on certification approaches for cybersecurity. It has pointed out the value of harmonizing certification practices for smart grid in Europe in one of their initial works in certification [19]. DIGITALEUROPE also supports this idea and encourages common security baselines with a standard set of guidelines for security levels and requirements [8]. Similarly, subsequent work on standardization from ENISA provides recommendations to standard development organizations to align with certification schemes [20] and review major available bodies working towards certification approaches [21].

2.3.6 ANSSI Security Classification

The ANSSI classification method is developed specifically for the security of ICS [4]. The management of different classes of ICSs has different requirements; essentially, the higher the class, the higher the impact and the security requirements. Based on various security parameters, general guidelines are proposed for determining a control system's security class. The ANSSI method aims to use as a basis an established Risk Analysis Method (e.g., EBIOS).

Figure 2.1 summarizes the ANSSI classification method. The likelihood is the result of combining three aspects: the *exposure*, the level of *accessibility* of ICS, and the level (or power) of *attackers*. Exposure is determined by combining the connectivity of ICS and the functionalities supported by the system.

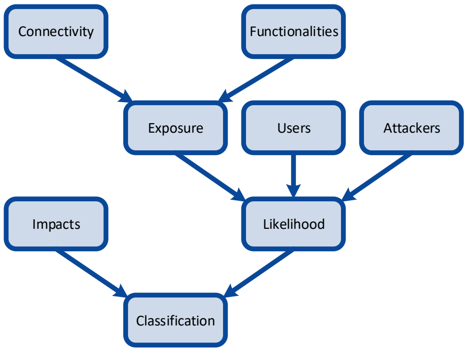


Figure 2.1: ANSSI Classification Method

2.3.7 BSI Kitemark for IoT

British Standards Institution (BSI) has also launched its Kitemark scheme targeting the manufacturers of IoT devices, and is based on the *Secure by Design* guidelines and measures provided by the government of the United Kingdom. To obtain the Kitemark, the manufacturers should first be compliant with ISO 9001[9]. In the Kitemark scheme, along with the functionality and interoperability testing, penetration testing and vulnerability scanning are also performed on the product. After the device obtains the Kitemark, the product should be regularly monitored and assessed. If flaws are identified, the Kitemark of the product

[8]https://www.digitaleurope.org/resources/digitaleuropes-views-on-cybersecurity-certification-and-labelling-schemes/

is revoked until it is fixed. BSI Kitemark for IoT products is categorized into residential (residential applications), commercial (commercial applications), and enhanced (high-risk applications).

Figure 2.2: BSI Kitemark for residential Internet of Things[9]

2.3.8 Underwriter Laboratories (UL) Security Rating

UL LLC has a long history as a global safety certification provider. They provide IoT security rating levels to measure the security of connected products. The major goals of this rating scheme are to help manufacturers and developers improve the security of their products, and rate the security to make it transparent and accessible to consumers. The security rating scheme provides five security levels: Bronze, Silver, Gold, Platinum, and Diamond [44]. Bronze is the lowest level containing the baseline security capabilities. Whereas Diamond is the highest security level having comprehensive security features. After the security capabilities are tested and evaluated, the security level is awarded. Figure 2.3 shows UL's IoT security rating levels. The number on the bottom-right of each mark is the identifier of the product. Using this number on the verification web page, one can find whether the product still holds the level or has been revoked.[10]

2.3.9 IoT Security Foundation (IoTSF)

IoTSF provides a checklist-based IoT security compliance framework [26] developed to guide industries to secure their IoT products through self-assessment, internal-certification, or certification by a third-party auditor. IoTSF also provides a tool implementation in excel to manage the questionnaires while applying the framework[12]. The process used in the compliance framework involves the following three steps:

Conduct risk analysis on the product in the target environment

This step helps create a risk register and determine security objectives for Confidentiality, Integrity, and Availability (CIA). Security objectives are assigned

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

[10] https://verify.ul.com/

[11] https://ims.ul.com/iot-security-rating-levels

[12] https://www.iotsecurityfoundation.org/wp-content/uploads/2018/12/IoTSF-Compliance-Questionnaire-Release-2.0-December-2018.xlsx

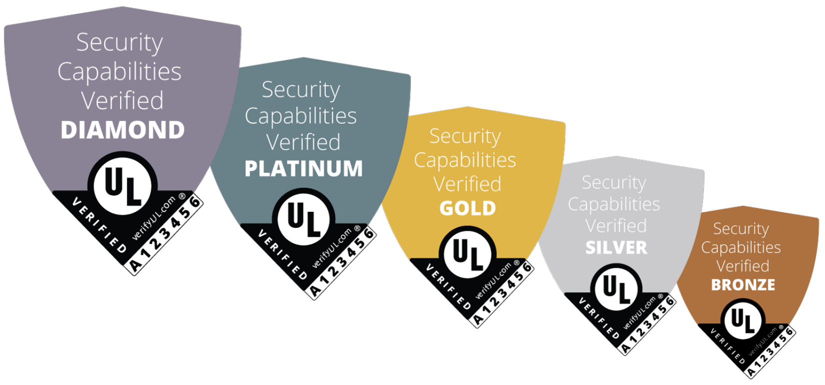

Security Capabilities Verified DIAMOND
Security Capabilities Verified PLATINUM
Security Capabilities Verified GOLD
Security Capabilities Verified SILVER
Security Capabilities Verified BRONZE
UL
VERIFIED
verifyUL.com
A123456

Figure 2.3: UL's IoT security levels [11]

as Basic, Medium, and High to the CIA based on the risk factor (impact x likelihood) computed for each threat.

Determine compliance class
Based on the security objective determined for CIA, the compliance class is determined. The tool provides the questionnaires related to the compliance class selected for the product.

Respond to each question
All questions generated should be addressed to comply with the given compliance class.

Figure 2.4 summarizes the steps for applying the IoT Security Compliance Framework.

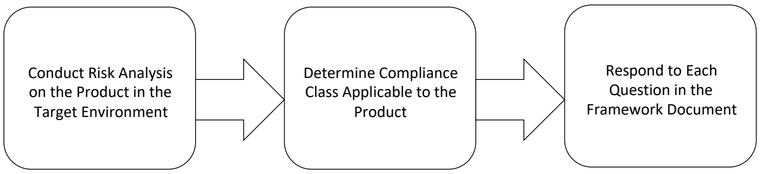

Conduct Risk Analysis on the Product in the Target Environment
Determine Compliance Class Applicable to the Product
Respond to Each Question in the Framework Document

Figure 2.4: Steps in IoT Security Compliance Framework

Global System for Mobile communication Association (GSMA) also provides a security assessment framework, which is also a checklist of security functionalities.

2.3.10 Summary

In this section, we have discussed the current standards and certification schemes towards IoT security. Standards can be compared to a checklist at an abstract level. Complying to a standard involves massive documentation and a considerable amount of resources, which requires a proper framework for managing all documents that make the process of compliance and audit easier. In this regard, managing compliance using the assurance approach, such as trust cases, can be effective. Also, certifying systems or products for a given standard is a long and expensive process, which is a major challenge. Therefore, we need a light-weight approach that can provide security assurance quickly and economically.

We have particularly focused the discussions on certification approaches to secure IoT systems. Certifications for cybersecurity are based on security tests that typically involve vulnerability scans and penetrations tests. In addition to attack-centric approaches such as vulnerability scans and penetration tests, it is also vital to ensure that the system is designed and implemented with a security mindset.

Other approaches that are meant for involving internal as well as third-party certification are checklist-based approaches where one has to specify which of the security requirements are addressed. These approaches do not provide mechanisms to reflect how well those requirements are fulfilled in the system.

2.4 Towards assurance cases

The goal of an assurance case is to obtain mutual acceptance of the subjective claims. Subjective claims are justified, producing good arguments with pieces of evidence to support the claims.

Properly structured arguments with evidence make the expert opinion explicit, resulting in better documentation of claims and improved communication between the experts. It can also help detect missing evidence and wrong assumptions made during decision making. Structured arguments are widely used in assurance cases [8, 40] to justify that what is being claimed is true. An assurance case is similar to a legal case where arguments are presented to support the claims backed by evidence [32]. Assurance cases are often used in safety-related domains such as the safety of nuclear power plants, railways, and medical devices.

Although safety and security often go hand in hand, there are certain differences between them. Thus, safety assurance methods cannot be directly ported into the security domain. Safety cases are more or less static and thus are manageable. However, in the case of security, it is hard to keep pace with evolving vulnerabilities and attacks. Moreover, adversaries are intelligent in the security realm and can take unexpected methods to attack the system [32]. Despite the differences between safety and security, adapting assurance cases in security is beneficial [2].

Safety cases have matured, being in use for decades in the safety domain. However, assurance cases in the field of security are relatively new and emerging. Sklyar and Kharchenko have introduced the framework for assurance case driven design for safety and security of IoT, where the safety and security features were assessed in the early stage and shown to improve the cost-effectiveness in the system life cycle [54]. Similarly, Mohammadi, Bunyadi, and Heisel proposed the trustworthiness cases to adapt the safety cases towards security to evaluate the trustworthiness of the system [46].

Security assurance for agile software development was also proposed, which involves mapping user stories to the security artifacts included as a (sub)claim in the security assurance case [50]. Thus, any changes in the code of a given user story result in reassessing the affected artifacts mapped to the user story.

2.4.1 Notations used in Assurance Cases

Representing the arguments by mere text is lengthy and monotonous, and it is not easy to persuade readers. Various diagrammatic notations such as Goal Structuring Notation (GSN) [55], Claim Argument Evidence (CAE) [8], and Toulmin argument model [57] can help experts structure and easily express their arguments. All the above notations structure the arguments as a tree where the root node is the main claim, which further grows into child nodes that provide the justifications using sub-claims and evidence. These methods are often used in constructing assurance cases. Several tool support implementations using a structured argument approach exist today to build assurance cases [45], where most of them comply with the GSN standard. GSN provides a set of symbols to represent the argument. Figure 2.5 summarizes the components of a GSN diagram.

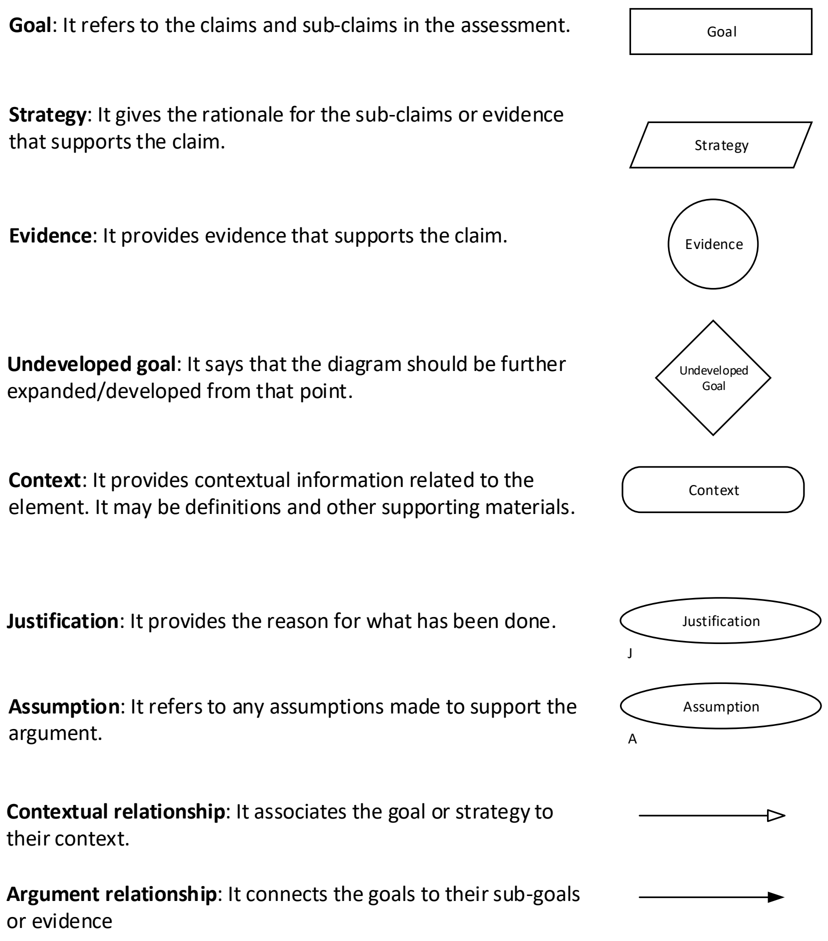
2.4.2 Confidence in Assurance Cases

Pieces of evidence typically support the claims in assurance cases. However, the evidence may not support the claim entirely and may have a degree of uncertainty. There are two major approaches to handling uncertainty in assurance cases: Bayesian Belief Networks (BBN) and Dempster-Shafer theory of evidence [12]. Applying BBN requires several initial inputs and demands some expertise in BBN and thus is not preferred if the user has to provide inputs. However, the Dempster-Shafer theory of evidence uses simple scales, and one can use it without much expertise.

In assessing assurance cases, each claim, sub-claim, and evidence have a degree of confidence that need to be aggregated to represent the overall confidence. Several approaches are proposed to compute the aggregated confidence [6, 12, 60]. The majority of methods propose a range of formulas to aggregate confidence based on how child arguments support the parent argument. According to a study on the confidence assessment in assurance arguments conducted in 2017, Graydon and Holloway concluded that the existing approaches of confidence in assurance cases are still imperfect and require further research [34].

2.4.3 Assurance Cases in Security

An assurance case is a tool to systematically structure the goals to convince that what is being claimed is correct. Various notations exist to help construct


Goal: It refers to the claims and sub-claims in the assessment.

Strategy: It gives the rationale for the sub-claims or evidence that supports the claim.

Evidence: It provides evidence that supports the claim.

Undeveloped goal: It says that the diagram should be further expanded/developed from that point.

Context: It provides contextual information related to the element. It may be definitions and other supporting materials.

Justification: It provides the reason for what has been done.

Assumption: It refers to any assumptions made to support the argument.

Contextual relationship: It associates the goal or strategy to their context.

Argument relationship: It connects the goals to their sub-goals or evidence

Figure 2.5: Summary of Goal Structuring Notation (GSN) symbols

assurance cases. Among them, the GSN is one of the most popular ones and is established as a standard.

Though there are differences between safety and security, assurance cases fit the security domain as well, and their use in the security domain is emerging. Some examples of their application are systematically extracting the security requirements, justifying the claim that the system is adequately secure, and confirm compliance towards a given standard.

Assurances cases can be static or dynamic. The dynamic assurance cases are used, for instance, in agile software development environments, where the part of the assurance case that requires re-assessment is identified based on the code changes. Using a security assurance case is suitable in different stages of the system development life cycle.

Assurance cases may involve subjective decisions, and thus it is crucial to reflect the confidence of the judgment made. The confidence can be improved by either producing well-formed arguments or by strengthening pieces of evidence. In spite of several applications of security assurance cases and approaches towards the computation of confidence, the propagation of confidence from the lower level arguments to the top-level arguments in the assurance case tree is still immature.

2.5 Baseline for research on secure IoT systems

We see that the majority of approaches towards security are attack-centric, where one focuses on the attacks and the attacker profile, and then proposes security measures to protect against them. It is a good idea to scan the system for vulnerability and patch them. However, building the system with a security mindset helps to reduce costs on the manufacturing side as well as the service providers side who use the IoT devices to provide customized solutions. One poor practice that commonly exists is that system's functionality is developed first, and security is added later after the vulnerabilities are found. This process can be costly in the long run.

There have been recent efforts towards the standards and certifications to secure IoT systems. They provide proper definitions and guidelines on what and how security should be done. Standards are typically targeted to a wide range of systems or products. They are usually abstract and not detailed. It requires experts to interpret and detail the specifications in the standard that is suitable to the context of the organization, and thus may be a concern for the organizations where security experts are lacking. Since standards list the significant areas to consider, it is always good to comply with them. However, because of the time to get certified and the cost of certification, it is infeasible for the low-cost IoT systems. There are also growing interests in labeling IoT products, and some organizations offer the labels based on the security evaluation and assessment. Again this is also an expensive process.

Until now, the existing approaches provide, at best, the set of guidelines to secure the system. However, not all IoT systems are meant to have the same level of security. Critical systems, e.g., health care and SCADA systems, may require a higher level of security. In contrast, IoT systems such as home control may require a moderate level of security. Therefore, the security of a system should be built based on the criticality of the system. There should be an approach that allows the selection of the security goals to which the given system should comply.

Thus, this research aims to have a goal-based light-weight method that is more feasible for IoT systems. To achieve our goal, we propose the security classification methodology, which builds on standards, certifications, and best practices to obtain the security criteria and functionality required to secure the system. We use these criteria to define security labels so that instead of fulfilling all the criteria specified in the standards, one can select certain security functionality to reach a given level of security. Also, we consider the domain and context of use of the application to define the security class. Like most methods in the security domain, our method is also not free from subjective judgments, and thus we adopt the concepts from the assurance cases to justify the evaluations using security arguments. To demonstrate the confidence over the claims and justifications made during argumentation, we also propose the confidence parameters using belief and uncertainty to the argument elements.

We further review available tools to support our methodology in practice but could not find any tool that could be used to implement our method as a toolchain. One of the closest tools to our needs is NOR-STA[13] because it supports the argument formulation and confidence handling. However, the mechanism of aggregating confidence does not align with our expectations. Thus, we implement a prototype of a tool that can be used to apply the security classification method.

[13]https://www.argevide.com/wp-content/uploads/2016/05/Argevide-NOR-STA-assurance-case.pdf

Chapter 3

Contributions

From the very beginning of the research, the main goal has been to involve non-security-experts in building secure systems. Our focus has been on the system designers and developers who play a major role in the design and development of the IoT systems. We first adapted the ANSSI classification method [4] and proposed a light-weight methodology for security classification for IoT. In particular, we tailored the methodology to fit the AMI systems. When we started this research, the smart meters were in the process of being deployed in Norway. Therefore, the early works used a different name, suggesting our original applications to smart grids. Later, we turned our focus towards the consumer IoT systems. We enhanced the classification methodology to define security criteria and applied the methodology to a real SHEMS. To demonstrate the applicability of our method, we evaluated the security class of the SHEMS and showed that the methodology could guide to improve the security class by making changes to the design or the security functionalities of the system.

3.1 LightSC: Light-weight Security Classification

We proposed the LightSC as the methodology for designing, analyzing, and evaluating the security of complex connected systems. We are motivated by offering the system/security engineers/designers a precise but light-weight method to guide their cybersecurity decisions while engineering critical infrastructure systems. To this end, the classes specified in our classification method provide "goals" to be reached by the engineer's designs, whereas our method of evaluating a system into a class is meant to provide guidelines for how to implement a system to meet the desired security. Note that traditional methods can be applied on top of our method if and when more value-driven evaluations are needed (thus, attacker models are factored in).

The LightSC methodology is built around three main factors: Connectivity, Security mechanisms, and Impacts. Connectivity reflects how the system is exposed to attacks. Security mechanisms reflect what security features are built into the system and what security functionalities are available, based on which the protection level is determined. Connectivity and protection level, when combined, form the Exposure. Figure 3.1 summarizes the security classification methodology.

In the classification method, we have considered five levels of connectivity (C):

- **C1**: Includes completely closed/isolated systems.

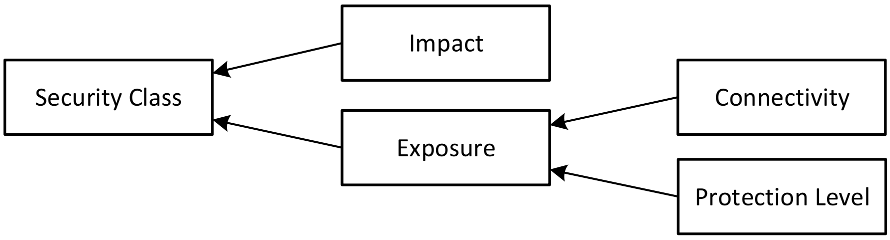


Figure 3.1: Methodology for computing security class

- **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., Virtual Private Network (VPN), private Access Point Name (APN), etc.). Allowing to access the corporate network only via VPN and allowing operators to connect to their field devices through their mobile device using a private APN are a few examples of C4.

- **C5**: Includes distributed systems with public infrastructure, i.e., like the C4 category except that the communication infrastructure is public (e.g. Web applications and services accessible using the Internet)

Similarly, there are five protection levels (P), which reflect the security mechanisms of the system. The protection level increases with the increasing number (or strength) of security mechanisms. Relevant security criteria should be defined to determine the protection level. Then for each criterion, the respective security mechanisms are derived. The security mechanisms are then grouped to form individual protection levels where higher protection levels have all the security functionalities of lower protection levels, including some additional and advanced functionalities. Protection level P1 represents no security mechanisms, whereas protection level P5 represents the strongest protection mechanisms. The evaluation of protection mechanisms is conducted by security experts. Exposure is then evaluated using protection level and connectivity. Table 3.1 (a) shows the evaluation of exposure from connectivity and protection level.

The impacts (or the consequences) have five levels: Insignificant, Minor, Moderate, Major, and Catastrophic. A security class is determined using impact and exposure. Table3.1(b) shows the lookup table for identifying a security class from exposure and impact levels.

Table 3.1: Calculations of (a) Exposure Levels and (b) Security Classes.

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

3.2 Confidence in LightSC method

The LightSC methodology involves a series of systematic steps to achieve a final security class. The majority of decisions made during each step are based on subjective judgments. Thus, we structure the assessment in LightSC as an argumentation model. Properly structured arguments with appropriate claims and evidence justifying the claims make the opinion explicit. This can also help to identify missing evidence and weak assumptions. The argumentation model can be represented as a tree structure using a notation such as the GSN where each claim is supported by its children sub-claims or evidence.

However, one may not always have full confidence in their decisions, i.e., accepting the justification and evidence. Confidence means the degree to which one agrees on the result of the assessment (belief) and the degree to which the evaluator lacks knowledge about the assessment (uncertainty). Thus, the evidence and claims are assigned a degree of belief. These beliefs are finally aggregated to compute the overall belief in the assessment. To support the aggregation, we consider that the nodes (claims and arguments) of the argument model have varying levels of importance for the class evaluation. Thus, we assign the importance using weights and adapt the multi-metrics approach proposed by Noll et al. to compute the aggregated belief [28, 48].

$$AggregatedBelief(Bel) = 100 - \sqrt{\sum_{i} \left(\frac{(100 - bel_i)^2 W_i}{\sum_i^n W_i} \right)} \tag{3.1}$$

where bel_i is the individual belief value of the component under consideration, and W_i is calculated from the component weight w_i as:

$$W_i = \left(\frac{w_i}{100} \right)^2 \tag{3.2}$$

In this way, the result of a LightSC assessment is represented using a three tuple <C, B, U>, where C is the class to which the system belongs, together with a belief measure B in the evaluation aspects of C, and the uncertainty U in the evaluation details. The uncertainty is calculated as the difference between plausibility and belief. We use the definition of belief and plausibility from the Dempster-Shafer theory [33], where plausibility is the upper bound of belief obtained by adding evidence to support the claim. For example, the evaluation <A, 84, 16> means that the result is class A with 84% confidence and 16% uncertainty. The 84% belief means that we have high confidence in the coverage of all necessary security measures to justify the protection level (P), exposure (E), and security class. An uncertainty of 16% indicates a moderate lack of justification for some of the arguments. One can tell how well the security assessment is justified by looking at the confidence parameters (belief and uncertainty).

3.3 LightSC methodology for DevSecOps

One major goal of the LightSC methodology is to involve system engineers and non-security-experts in building secure systems. This requires that our methodology can be used within a standard software development life cycle. In particular, we want that the LightSC methodology fits the DevSecOps culture, which is growing in popularity. DevSecOps is an extension of DevOps with security aspects and tools throughout all the software development life cycle stages. However, classical security classification methods are highly manual and thus slow, not fitting the rapid DevSecOps cycles.

To make the LightSC methodology DevSecOps ready, we identified five principles, namely (1) dynamicity, (2) tool-based, (3) easy to use, (4) static impact, and (5) oriented on protection mechanisms. As LightSC already fulfills the principles (4) and (5), we also worked so that it fulfills principles (2) and (3) by developing a usable tool to support the LightSC methodology and performing a usability evaluation of the tool.

The development of a Security Classification Tool (SCT) involved multiple stages of prototyping and usability testing. The final version of the tool was implemented as a web application. The development and enhancement of the tool are based on the feedback from the evaluation stages, which involved questions and observations. The SCT prototype was developed as an Asp .Net Core MVC application with SQL database for persistence. We used ASP.NET Core Identity to manage authentication and authorization in the application [1]. For responsive user interfaces, we used bootstrap. The tool is deployed as a Microsoft Azure App Service at *https://lightsc.azurewebsites.net/*.

The tool supports four main functionalities: define the system, add the components to the systems, perform the assessment, and compute class. The first three steps require manual inputs from the user, with the final result being the automatically computed security class.

1. **Define System**

   A user should first define a system for which to compute the security class. Here the user gathers details about the system and gives a name to the

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

system. The details may include how the system works, which technology it uses, and which components exist.

2. **Add components**

   In the security classification method, a system is decomposed into its components. Thus, after the system is defined, its components should be defined. The description of the component may include information such as the role of the component, vendor information and communication standards used. One may also include communication capabilities and scenarios where the component is used and how it interacts with other system components. The description provides an overview of the component to less expert personnel. It also includes the selection of the type of the component. In our case, we evaluated IoT systems, and thus, examples of component types may include IoT devices, IoT Hub, and Backend System. One can also define their own component type. The user is also required to define the connectivity and the impacts of a security breach.

3. **Perform assessment**

   Here, the user should select the security functionalities present in the system to determine the protection level. The user has to select the relevant criteria and assign the weights to them based on their significance. From the relevant criteria, the user then selects the security functionalities that are in place. The user also assigns the weight to the security functionality and provides the belief value to show the confidence he/she has in the existence of the functionality. The belief value should be justified by the evidence supporting the security functionality.

4. **Compute class**

   After all the inputs are provided to the components, the user can then compute the security class. The final class is automatically calculated by the application based on inputs provided by the user. The security criteria and security functionalities are used to compute the protection level, exposure, and security class for the given component. Similarly, the computed protection level and connectivity information from the user are used to compute exposure. Finally, exposure and the impact level provided by the user can be used to compute the security class. The beliefs and weights are also assigned by the user in step 3, which is used to compute the aggregated belief in the assessment.

Besides the above core functionalities, the user can also save the assessment for future reference. The tool should allow users to browse the assessment and perform CRUD (Create, Read, Update, Delete) operations on system and system components.

3.4 Evaluation

The validation of the methodology was done by conducting extensive case studies in the domain of AMI and SHEMS. After the prototype was developed, several stakeholders validated the methodology by applying it to IoT systems from their organizations. In this section, we discuss the case studies that we performed and discuss the applicability of the methodology after being used by the stakeholders.

3.4.1 Case I: AMI System

The smart grid embraces the power of IoT to establish the next-generation power grid. It uses a two-way flow of both electricity and information to create a widely distributed automated energy delivery network [24]. Smart grids form one of the largest networks as they connect every end node (houses, industries, buildings, etc.) having access to electricity. Along with increasing connectivity, a smart grid collects massive amounts of data, which may contain critical and private information. Protecting such complex infrastructures from cyber-physical attacks is a serious concern.

When we started the research, the deployment of smart meters was already in progress in Norway. Being aware of the security challenges of such systems, we investigated the topology of ongoing AMI installations in detail. In the meantime, Hansen, Staggs, and Shenoi [36] also studied AMI deployments in the United States, describing AMI infrastructure components and attack surfaces. These authors identified potential attacks and impacts of attacks on AMI, which can then be used in risk assessment. Their work concludes by describing the lack of a unified framework for analyzing the security aspects of complex smart grid systems. This was our main motivation towards proposing a methodology to analyze impacts and exposure aspects of complex smart grid systems.

In the initial phase to apply the security classification methodology, security experts should analyze the system and identify all the security criteria and respective security functionalities. This may require the involvement of non-security-experts who understand the system well. The security experts then define the protection levels by appropriately grouping the security functionalities into different levels. Finally, the lookup table for identifying exposure and security class is constructed. After this, the security classification is ready to be used by non-experts where they can use the template created by the experts to evaluate protection level, exposure, and security class to (i) evaluate and/or (ii) design the system.

The methodology is functionality-centric and goal-centric, where one focuses on the security functionality, connectivity, and impact of attacks to set a goal class or evaluate the security class. For new systems, extensive threat analysis may be essential to determine the security requirements. However, for systems belonging to a domain whose security analysis has been performed and security functionalities have been identified, it is straightforward to apply the security classification methodology.

In the case study of AMI systems, the system was comparatively new, and thus the system architecture was broken down into sub-systems, components, and communication mechanisms, which were then analyzed. We also defined the security criteria and extracted the security functionality relevant for each security criteria. The impact, connectivity, and exposure aspects of the system were also discussed in detail to show how the security classification methodology is applicable to evaluate the system security of existing AMI systems. In this study, we identified security aspects that need to be considered when applying security classification to an AMI, focusing on the details from the Kamstrup AMI installations in Norway.

3.4.2 Case II: SHEMS

The SHEMS can be seen as the integration of IoT devices that allow sensing, measuring, and controlling the electric appliances to provide energy management services such as demand response programs to the consumers [43]. A SHEMS is dedicated to saving energy by monitoring and managing electrical appliances, including load, storage, or generation resources [42]. Heat pumps are typical examples of load resources. Similarly, car batteries and solar panels are examples of storage and generation resources, respectively. Functional modules of SHEMS may include monitoring, logging, control, management, or alarm services [61]. After our initial AMI case study, we moved to consumer IoT systems and applied the security classification method to an existing SHEMS from E2U Systems AS.

We initially did not have any template that the non-experts could use to apply the security classification methodology. Thus, we analyzed the selected SHEMS to understand its architecture, components, and working mechanisms. We then mapped the reference architecture of the SHEMS with the one proposed by Ghirardello et al. [29] and described the components of SHEMS as IoT devices, IoT hub, residential gateway, communication channels, backend system, and application & network data. We referred to the standards and best practices to extract the security criteria and security functionalities. We then grouped the security functionalities to categorize five different protection levels. Finally, two lookup tables were constructed to determine the exposure and security class.

After the template for evaluation was ready, we first evaluated the security class of the SHEMS in the current state as class D because of major impact, higher connectivity (C5), and moderate security mechanisms. This was not considered a suitable class. The lookup table indicated that either reducing impact or exposure could improve the security class. Similarly, reducing connectivity or increasing protection level could reduce the exposure. The connectivity could be reduced using an edge controlling mechanism instead of a centralized controlling mechanism, minimizing both impact and exposure with an achievable security class A. Thus, using our methodology, one can indicate how the system needs to be improved to obtain an acceptable security class.

3.4.3 Case III: Case studies and usability evaluation by partners

The idea behind the tool support was to make the methodology fit within the tool-chains and development philosophy of modern system development life cycle (in our case DevSecOps). The requirement of the DevSecOps demands the methodology to be supported by a tool that is also user-friendly. Therefore, we decided to evaluate the fulfillment of these requirements through user evaluations. Our target groups included both individuals and teams with diverse expertise, which is essential in usability testing to cover well the target user group. Our evaluators included real stakeholders (i.e., the SCOTT project partners), master's students from one course in measurable security, Small and Medium-sized Enterprises (SME) from a Polish cluster, and individuals from software development industry. The development of the LightSC involved multiple stages of prototyping and usability testing.

We first translated the methodology into a ten-step process followed by implementing a low-fidelity prototype in spreadsheets. After that, we created a high-fidelity prototype as a web application, which was further enhanced, and the second version of the prototype was also produced. All the stages utilized the user-centric approach and involved several rounds of workshops, webinars, user observation, user feedback, and surveys with the evaluators.

The evaluation from the final version of the LightSC suggests that the tool is easy enough to be used by non-security-experts. It encouraged us to release it as a public tool. The more experimental 'beliefs and weights' part of the tool was considered complex to use.

In total, throughout all our stages of creating the tool, we saw the LightSC applied to 19 different IoT systems, done mostly by non-security-experts or teams, using our different prototype implementations. These provided valuable feedback regarding the usability of the LightSC prototypes that we have been building, but can also be seen as valid proofs of the applicability of the original security classification methodology that we have proposed.

Chapter 4

Overview of the Research Papers

This thesis is built upon the scientific contributions made by four scientific papers. Paper 1 describes the challenges of traditional security approaches in IoT and proposes the concept of security classes that fit complex systems such as smart grids. Paper 2 extends the security classification methodology from Paper 1 and shows the applicability of the methodology in SHEMS (these are systems that are usually part of smart grids but also part of smart homes and thus more close to consumer IoT products than AMI systems are). Based on lessons learned from Paper 2, the notion of confidence in security classification methodology is proposed in Paper 3. Finally, in Paper 4, the tool to support security classification is proposed. The tool support is evaluated for usability by the stakeholders so as to comply with DevSecOps principles. This chapter summarizes the contributions made by these four papers.

4.1 Paper 1: A Methodology for Security Classification applied to Smart Grid Infrastructures

In this paper, we investigated the topology of AMI in detail, which was already under deployment in Norway, to look into the security aspects of AMI. We then investigated the current practices and ongoing activities towards the security of smart grid systems to point out the problem in current approaches. Using the knowledge obtained from such approaches, we proposed a new methodology to fill the gap.

AMI is considered critical infrastructure and forms one of the largest networks producing massive amounts of data that may contain critical and private information. The *two-way* flow of information in AMI or smart grid is meant to make the electricity network more efficient and reliable. These infrastructures are driven by IoT, which in addition to creating novel services, opens up new attack vectors. AMI is relatively new, and it lacks a proper framework to address security challenges. We list the following challenges in the security of critical IoT systems.

- **Only attack-centric approaches are not enough.**

  New vulnerabilities are discovered all the time, and systems cannot be protected from all unknown threats which might come up in the future. Thus, in addition to attack-centric approaches, we need to group our security model after the security priorities, where the system at the design time is targeted to a given security level based on the context of use and features it supports.

- **Current approaches are heavy in terms of cost and time.**

> Traditionally, organizations look at standards or certification approaches when selecting the right product. In the case of IoT, there are no such standards readily available, though there have been efforts towards standardization such as ETSI TS 103 645, ISO/IEC 27030, and labeling of IoT products such as BSI kitemark, and ENISA baseline security recommendations. Certification, such as Common Criteria, takes a long time to get certified and is expensive. Since IoT devices are designed and manufactured as low-cost systems and change frequently, such approaches would not be appropriate.
>
> In organizations using services provided by IoT such as utilities, risk assessment is a common approach. It requires significant involvement of experts to perform the assessment and also takes time to be completed. Though risk assessment is an essential step for such organizations, we also need a fast and light-weight approach that, in addition to security-experts, also allows participation of non-security-experts in the assessment. Such frameworks can also help utilities set the baseline security goal and decide which product to select to maximize the security of their system, keeping the cost under consideration.

This paper proposes a security classification methodology to address the aforementioned challenges. Unlike traditional approaches such as risk analysis methods, which consider attackers and likelihood a vital part of the evaluation, our security classification methodology focuses on the exposure and security functionality aspects of complex IoT systems. This security classification method offers a concrete approach to guide security decisions by system designers to meet a desired security goal.

After proposing the methodology, we tailored and applied it to AMI. Using the details of AMI components and the interactions between them, we pointed out possible areas to look into while making design decisions. Using available standards and best practices for IoT, we defined the protection criteria for the AMI. Then for each component, we described the exposure and relevant security criteria.

This paper also showed how the proposed classification methodology fits the AMI context. In this work, we did not evaluate the AMI against security classes because the details of the security functionality for the selected AMI could not be obtained. However, we showed the relevant security criteria applicable to AMI components and possible exposure and considerations to evaluate each parameter of the security class. The methodology is primarily targeted to system designers and developers to guide them in making security decisions. However, the notion of security class is also relevant to regulatory bodies to enforce security regulations, vendors and service providers to build and select secure systems, and end-users to select IoT systems with adequate security.

4.2 Paper 2: Criteria for Security Classification of Smart Home Energy Management Systems

This paper extends the methodology proposed in Paper 1 towards SHEMS and systematically applies it to an existing SHEMS. It describes the architecture of a typical SHEMS, based on which, the security criteria and the protection mechanisms per criteria using the standards and guidelines were extracted. The security of existing SHEMS from E2U Systems was analyzed using the security classification methodology. In particular, the device control command was selected for analysis. The SHEMS had a centralized architecture where the backend system selects and sends control signals to the selected devices. The security class evaluation for this system resulted in class D (Connectivity C5, Impact Major, Protection Level P4), which was considered unsuitable. An alternative design was then proposed, where the logic for sending control signals persisted in the gateway instead of the backend system, which reduced the connectivity. This design change also reduced the impact, and the resulting class progressed to class A. This paper showed that the system designers could use our methodology as a catalog to pick up the security features to design the system targeted to a given class.

Until now, the capability and applicability of the security classification method were shown. Since the methodology heavily relied on expert judgment, the credibility of the evaluation was not addressed. Thus, future work was to provide assurance in the evaluation. Further, the aggregation mechanism to evaluate the overall class of the system remained an issue.

4.3 Paper 3: Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT

This paper attempts to improve the trustworthiness of the proposed security classification methodology. Like most other approaches in security, our security classification method also involves expert judgment. However, it does not provide any concrete justification for the decisions, resulting in lower trust in the assessment. We proposed to build confidence in the security class evaluation using beliefs and arguments. The use of arguments is common in safety domains. In this work, we first build arguments for claims made during the evaluation. We then justify each claim by producing adequate evidence to support it. Based on the evidence produced, the confidence is computed. The overall class evaluation can be constructed as a tree structure using tools such as the GSN diagram.

The final class is represented along with the confidence parameter as a tuple. Here the tuple A, 84, and 16 means that the result obtained is class A with 84 percent belief and 16 percent uncertainty. Belief shows the percentage of confidence earned from the available evidence. If the evidence is missing or weak, the belief is weak.

To reflect the confidence parameter, we assign the belief and plausibility parameters to the evidence and sub-claims. To obtain the overall confidence, we aggregate these using the root mean squared data as in the multi-metrics. We considered the root mean squared approach because they tend to aggregate the beliefs realistically than aggregating the mean value.

4.4 Paper 4: Security Classification for DevSecOps: Five Principles and an Exemplification

This paper discusses the fact that the security classification methodology is still highly manual and thus requires to improve its usability by automating several parts of the methodology. It proposes the initiative towards developing tool support for applying security classification methods to fit the DevSecOps culture. The paper defines five major principles to guide security classification methodologies to be DevSecOps compatible and shows that our security classification methodology implemented as a tool fulfills those principles. The tool implementation followed the systematic interaction design process, where the usability of the tool was also evaluated using the real stakeholders.

In this work, we first simplified the security classification methodology by translating it into a series of steps and discussing the requirements for a tool to support the methodology. Then we systematically discuss the development of the user requirements by using the initial low-fidelity spreadsheet version of the prototype followed by a high-fidelity prototype as a web application. In particular, we used the spreadsheet implementation of the tool to gather users' expectations and needs. Then the web version of the tool was implemented, following up on the feedback from the users. Finally, the second version of the web application was also implemented based on the feedback on the first version. The second version was then put to larger evaluations to validate the tool support of the security classification methodology. Thus, in this paper, the applicability of the tool was validated by using it to evaluate a considerable number of real IoT systems. The usability test of the tool was also conducted with stakeholders to validate the concept and gain the user needs to be able to use this tool.

Chapter 5

Conclusion

This chapter summarizes our research and discusses the limitations and future directions that this research can continue.

5.1 Summary of Contributions

We first proposed the notion of security class, which unlike most methods that are attack-centric, is functionality-centric and system-centric, where one can select the appropriate security mechanisms for systems or system components to reach a desired security class. Instead of reinventing the wheel, the methodology can go hand in hand with current risk assessment approaches. The assessment in our methodology is useful for risk assessment as well. Next, to validate the methodology, we analyzed real Advanced Metering Infrastructure deployments and Smart Home Energy Management System to show the applicability of our methodology both to design and evaluate the security of IoT systems. We further enhanced the methodology by introducing the confidence in the evaluation, which uses concepts from the assurance cases where instead of assuring the adequacy, we claim for the confidence of the subjective decision. To further demonstrate the applicability and simplify the methodology for non-security experts (i.e., system designers and developers), we outlined five principles for a security classification to be DevSecOps compatible. To satisfy these principles, we implemented a tool to support the methodology. It enabled and resulted in the participation of several stakeholders who applied the security classification methodology to their real systems to support the validation. Through the usability evaluations, we received valuable feedback, which helped us improve the methodology as well as the user-friendliness of the implementation of the security classification methodology tool.

5.2 Answers to the research questions

RQ1: How can we ease the creation of secure IoT systems?

This question is first answered in Paper 1 by proposing a light-weight security classification methodology to support self-assessment and design of secure systems. Paper 1 first describes the major issues in IoT security and proposes the core idea that the security of IoT systems can be measured by connectivity, applied protection mechanisms, and impacts. Based on these parameters, we defined the concept of security classification. The consecutive papers (Paper 2 and Paper 3) are continuous enhancements of the methodology and validations with more case studies. The applicability of the methodology in Paper 2 and Paper 3 is demonstrated using the case study of a SHEMS. Finally, Paper 4 proposes the tool support that makes it easy for users to apply our methodology.

RQ2: How can we measure the trustworthiness of the proposed method?
Paper 3 answers this question by introducing the structured argument, belief, and uncertainty in the security classification. Subjective opinions during the assessments are made explicit, using structured arguments. Based on the evidence presented to support the arguments, one could determine the confidence parameters for each of the claims and sub-claims. The aggregated confidence parameters represent the measurable overall trustworthiness of the security class assessment.

RQ3: How can we enable system engineers and non-security-experts to evaluate the security of systems and provide them with solutions being compliant with the envisaged security goals?
This question is answered in Paper 1, Paper 2, Paper 3, and Paper 4. The primary goal of this thesis is to reduce the dependency on security experts in the security process. Paper 1 describes the concept. Paper 2 shows the applicability where we apply the methodology to evaluate the security and make use of the result to improve the security class. Similarly, Paper 3 also shows the applicability of the enhanced methodology. Finally, in Paper 4 we focused on the system development life cycle and proposed a tool to show that the LightSC methodology is compatible with the DevSecOps culture. We evaluated the methodology by involving real stakeholders who were not security experts. In all these works, we have defined security criteria and demanded that someone who uses the methodology should have a basic security understanding but not necessarily be an expert. We have shown that by having a good understanding of the system, one can use our methodology to select the security goal and verify this based on the security features that the system possesses, without performing actual security tests. Verification could be done using the documents provided by vendors and service providers of IoT systems that describe the security features of the system and system components. Security experts can also use this methodology to strengthen confidence by providing evidence to support the claims through the experimental results.

RQ4: How can a tool simplify the use of the proposed methodology?
Paper 4 answers this question by proposing tool support to apply the methodology. Before Paper 4, applying the methodology was highly manual and required considerable effort from the user. In this paper, the security classification methodology was simplified, defining the step-by-step tasks that the user should follow to apply the methodology. These steps were further reduced using the tool. We performed the evaluation and the usability test of the tool using real stakeholders as participants. Also, to make the tool practically usable and become a part of the Software Development Life Cycle (SDLC), we made the tool DevSecOps compliant so that it can be motivating and simple to use by the system designers and developers. The proposed tool support was evaluated by real stakeholders as well as professionals in the software industry to support our

claim that our methodology reduces the time, effort, and expertise for applying the SC methodology. The evaluation results showed that the tool support indeed helps simplify the LightSC methodology and is moderately easy to use.

5.3 Limitations

In our research, most case studies use real systems to show the applicability and feasibility of security classification methodology. However, we have not applied the methodology to the whole system to obtain an overall class.

Though we talked about structuring the arguments and supporting them through evidence, we have not implemented these features in our tool support for LightSC methodology. Implementation of such features could have helped to validate our concept in real systems further.

In the LightSC methodology, we have not considered the whole life cycle of IoT systems. For instance, criteria such as decommissioning the IoT devices, which also determines the security of the IoT systems, are not considered. Security experts can include newer criteria in the initial phase of the classification method to define the protection levels if required.

5.4 Future work

One immediate task that can be done is to implement the missing features mentioned in the limitation section into the prototype to investigate the usability of such features. In our work, we have claimed that the concept of security classes is useful for regulatory bodies. However, we have not investigated details of how the methodology of security classification can be put together with the regulatory bodies and the IoT manufacturers so that both parties cooperate towards securing IoT systems.

Because our methodology aims to reduce the dependency on security experts to build secure systems, we see the opportunities for our methodology to be further expanded in a secure system development life cycle, particularly in the DevSecOps cycle. One scenario of its use is to extract the security requirements based on the goal security class for a system when the functional requirements are developed. The security requirements can then be pulled down to the backlogs level. The security assurance made in each backlog can also be propagated towards the system level, where the system security class is demonstrated. It would be interesting to see how the security classification methodology can be integrated into the DevSecOps pipeline and how system security is dynamically demonstrated in real-time.

The current security classification is still manual. However, there is an opportunity for LightSC to be dynamic and automatic in terms of requirement generation and validation. It is worth investigating mechanisms of centralizing the security requirements, where based on the new threats, the security requirement and protection level get updated, and the parties utilizing the dynamic evaluation scheme are benefited without having to update their class requirements when things start to change.

We also found from the evaluation of our methodology that the majority of non-security-experts found it difficult to assign beliefs and weights to the parameters, and none of the users seemed to use those parameters. Thus, there is room for the opportunity to investigate the ways to assign the weights and beliefs automatically. We see that, though these values are subjective, during the analysis/initial phase, weights can be pre-assigned to many high-level properties such as security criteria and some of the security functionalities as well. These weights can be based on the domain of the system, such as the priority of some security functionalities in critical systems, which could be different from that of non-critical systems. Similarly, if we have the concept of centralized requirements discussed in the previous paragraph, beliefs can also be semi-automated, at least in the security criteria level. However, to be flexible towards expert users, these parameters should be able to be changed if one wants to.

Bibliography

[1] Alberts, C. et al. *Introduction to the OCTAVE Approach.* Tech. rep. Carnegie-Mellon University, Software Engineering Institute, 2003.

[2] Alexander, R., Hawkins, R., and Kelly, T. "Security assurance cases: motivation and the state of the art". In: *High Integrity Systems Engineering Department of Computer Science University of York Deramore Lane York YO10 5GH* (2011).

[3] Anderson, R. and Fuloria, S. "Certification and evaluation: A security economics perspective". In: *2009 IEEE Conference on Emerging Technologies & Factory Automation.* doi: 10.1109/ETFA.2009.5347129. IEEE. 2009, pp. 1–7.

[4] ANSSI. "Classification Method and Key Measures". In: (2014).

[5] Asghari, P., Rahmani, A. M., and Javadi, H. H. S. "Internet of Things applications: A systematic review". In: *Computer Networks* vol. 148 (2019). doi: 10.1016/j.comnet.2018.12.008, pp. 241–261.

[6] Ayoub, A. et al. "Assessing the overall sufficiency of safety arguments". In: (2013).

[7] Baldini, G. et al. "Security certification and labelling in Internet of Things". In: *IEEE 3rd World Forum on Internet of Things (WF-IoT).* doi: 10.1109/WF-IoT.2016.7845514. IEEE. 2016, pp. 627–632.

[8] Bishop, P., Bloomfield, R., and Guerra, S. "The future of goal-based assurance cases". In: *Proc. Workshop on Assurance Cases.* Citeseer. 2004, pp. 390–395.

[9] Blythe, J. M., Sombatruang, N., and Johnson, S. D. "What security features and crime prevention advice is communicated in consumer IoT device manuals and support pages?" In: *Journal of Cybersecurity* vol. 5, no. 1 (2019). doi: 10.1093/cybsec/tyz005, tyz005.

[10] Bojanova, I. and Voas, J. "Trusting the internet of things". In: *IT Professional* vol. 19, no. 5 (2017). doi: 10.1109/MITP.2017.3680956, pp. 16–19.

[11] Celebucki, D., Lin, M. A., and Graham, S. "A security evaluation of popular Internet of Things protocols for manufacturers". In: *ICCE.* doi: 10.1109/ICCE.2018.8326099. IEEE. 2018, pp. 1–6.

[12] Cyra, L. and Górski, J. "Support for argument structures review and assessment". In: *Reliability Engineering & System Safety* vol. 96, no. 1 (2011). doi: 10.1016/j.ress.2010.06.027, pp. 26–37.

[13] Department for Digital, Culture, Media & Sport. *Code of Practice for Consumer IoT Security*. Standard. url: https://assets.publishing.service.gov.uk/-government/uploads/system/uploads/attachment_data/file/773867/Code_-of_Practice_for_Consumer_IoT_Security_October_2018.pdf. Government of UK, 2018.

[14] Department for Digital, Culture, Media & Sport. *Secure by Design: Improving the cyber security of consumer Internet of Things Report*. Report. url: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/775559/Secure_by_Design_Report_.pdf. Government of UK, 2018.

[15] Dey, V. et al. "Security vulnerabilities of unmanned aerial vehicles and countermeasures: An experimental study". In: *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*. doi: 10.1109/VLSID.2018.97. IEEE. 2018, pp. 398–403.

[16] ENISA, E. *Appropriate Security measures for smart grids*. url: https://www.enisa.europa.eu/publications/appropriate-security-measures-for-smart-grids/at_download/fullReport. Dec. 2012.

[17] ENISA, E. *Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures*. url: https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot/at_download/fullReport. Nov. 2017.

[18] ENISA, E. *Good Practices for Security of IoT - Secure Software Development Lifecycle*. url: https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot-1/at_download/fullReport. Nov. 2019.

[19] ENISA, E. *Smart Grid Security Certification in Europe*. url: https://www.enisa.europa.eu/publications/smart-grid-security-certification-in-europe/at_download/fullReport. Dec. 2014.

[20] ENISA, E. *Standardisation in support of the Cybersecurity Certification*. url: https://www.enisa.europa.eu/publications/recommendations-for-european-standardisation-in-relation-to-csa-i/at_download/fullReport. Feb. 2020.

[21] ENISA, E. *Standards Supporting Certification*. url: https://www.enisa.europa.eu/publications/recommendations-for-european-standardisation-in-relation-to-csa-ii/at_download/fullReport. Feb. 2020.

[22] ETSI. *Cyber Security for Consumer Internet of Things*. Standard ETSI TS 103 645 V1.1.1. url: https://www.etsi.org/deliver/etsi_ts/103600_103699/103645/01.01.01_60/ts_103645v010101p.pdf. European Telecommunications Standards Institute, 2019.

[23] ETSI. *CYBER; Cyber Security for Consumer Internet of Things: Baseline Requirements.* Standard ETSI EN 303 645 V2.1.0. url: https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.00_30-/en_303645v020100v.pdf. European Telecommunications Standards Institute, 2020.

[24] Fang, X. et al. “Smart grid – The new and improved power grid: A survey”. In: *IEEE Communications Surveys & Tutorials* vol. 14, no. 4 (2012). doi: 10.1109/SURV.2011.101911.00087, pp. 944–980.

[25] Fouladi, B. and Ghanoun, S. “Security evaluation of the Z-Wave wireless protocol”. In: *Black hat USA* vol. 24 (2013). url: http://www.neominds.org/download/zwave_wp.pdf, pp. 1–2.

[26] Foundation, I. S. *IoT Security Compliance Framework.* Tech. rep. url: https://www.iotsecurityfoundation.org/wp-content/uploads/2018/12/IoTSF-IoT-Security-Compliance-Framework-Release-2.0-December-2018.pdf. IoT Security Foundation, Dec. 2018.

[27] Fredriksen, R. et al. “The CORAS framework for a model-based risk management process”. In: *International Conference on Computer Safety, Reliability, and Security.* Ed. by Anderson, S., Felici, M., and Bologna, S. doi: 10.1007/3-540-45732-1_11. Springer. 2002, pp. 94–105.

[28] Garitano, I., Fayyad, S., and Noll, J. “Multi-metrics approach for security, privacy and dependability in embedded systems”. In: *Wireless Personal Communications* vol. 81, no. 4 (2015). doi: 10.1007/s11277-015-2478-z, pp. 1359–1376.

[29] Ghirardello, K. et al. “Cyber security of smart homes: Development of a reference architecture for attack surface analysis”. In: (2018). doi: 10.1049/cp.2018.0045.

[30] Gilsinn, J. D. and Schierholz, R. *Security assurance levels: a vector approach to describing security requirements.* Tech. rep. 2010.

[31] Glass, R. L. “A structure-based critique of contemporary computing research”. In: *Journal of Systems and Software* vol. 28, no. 1 (1995), pp. 3–7.

[32] Goodenough, J., Lipson, H., and Weinstock, C. “Arguing security-creating security assurance cases”. In: *rapport en ligne (initiative build security-in du US CERT), Université Carnegie Mellon* (2007).

[33] Gordon, J. and Shortliffe, E. H. “The Dempster-Shafer theory of evidence”. In: *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project* vol. 3 (1984), pp. 832–838.

[34] Graydon, P. J. and Holloway, C. M. “An investigation of proposed techniques for quantifying confidence in assurance arguments”. In: *Safety science* vol. 92 (2017). doi: 10.1016/j.ssci.2016.09.014, pp. 53–65.

[35] Greveler, U. et al. “Multimedia content identification through smart meter power usage profiles”. In: *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*. WorldComp. 2012, pp. 1–8.

[36] Hansen, A., Staggs, J., and Shenoi, S. “Security analysis of an advanced metering infrastructure”. In: *International Journal of Critical Infrastructure Protection* vol. 18 (2017). doi: 10.1016/j.ijcip.2017.03.004, pp. 3–19.

[37] Hossain, M. M., Fotouhi, M., and Hasan, R. “Towards an analysis of security issues, challenges, and open problems in the internet of things”. In: *2015 IEEE World Congress on Services*. doi: 10.1109/SERVICES.2015.12. IEEE. 2015, pp. 21–28.

[38] Jones, J. *Factor analysis of information risk*. US Patent App. 10/912,863. Aug. 2004.

[39] Keblawi, F. and Sullivan, D. “Applying the common criteria in systems engineering”. In: *IEEE security & privacy* vol. 4, no. 2 (2006). doi: 10.1109/MSP.2006.35, pp. 50–55.

[40] Kelly, T. “Reviewing assurance arguments-a step-by-step approach”. In: *Workshop on assurance cases for security-the metrics challenge, dependable systems and networks (DSN)*. 2007.

[41] Klein, G. et al. “Formally verified software in the real world”. In: *Communications of the ACM* vol. 61, no. 10 (2018). doi: 10.1145/3230627, pp. 68–77.

[42] Lee, J. I. et al. “A study on the use cases of the smart grid home energy management”. In: *ICTC*. doi: 10.1109/ICTC.2011.6082716. IEEE. 2011, pp. 746–750.

[43] Liu, Y. et al. “Review of smart home energy management systems”. In: *Energy Procedia* vol. 104 (2016). doi: 10.1016/j.egypro.2016.12.085, pp. 504–508.

[44] LLC, U. *Methodology for Marketing Claim Verification: Security Capabilities Verified to level Bronze/Silver/Gold/Platinum/Diamond*. Tech. rep. UL LLC, 2019.

[45] Maksimov, M. et al. “Two decades of assurance case tools: a survey”. In: *International Conference on Computer Safety, Reliability, and Security*. doi: 10.1007/978-3-319-99229-7_6. Springer. 2018, pp. 49–59.

[46] Mohammadi, N. G., Ulfat-Bunyadi, N., and Heisel, M. “Trustworthiness Cases–Toward Preparation for the Trustworthiness Certification”. In: *International Conference on Trust and Privacy in Digital Business*. doi: 10.1007/978-3-319-98385-1_17. Springer. 2018, pp. 244–259.

[47] Molina-Markham, A. et al. “Private Memoirs of a Smart Meter”. In: *Proceedings of the 2^{nd} ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*. BuildSys’10. doi: 10.1145/1878431.1878446. Zurich, Switzerland: ACM, 2010, pp. 61–66.

[48] Noll, J. et al. “Measurable security, privacy and dependability in smart grids”. In: *Journal of Cyber Security and Mobility* vol. 3, no. 4 (2014). doi: 10.13052/jcsm2245-1439.342, pp. 371–398.

[49] Nurse, J. R., Creese, S., and De Roure, D. “Security risk assessment in Internet of Things systems”. In: *IT professional* vol. 19, no. 5 (2017). doi: 10.1109/MITP.2017.3680959, pp. 20–26.

[50] Othmane, L. B. and Ali, A. “Towards effective security assurance for incremental software development the case of zen cart application”. In: *2016 11th International Conference on Availability, Reliability and Security (ARES)*. doi: 10.1109/ARES.2016.86. IEEE. 2016, pp. 564–571.

[51] Redmill, F. “Understanding the use, misuse and abuse of safety integrity levels”. In: *Proceedings of the Eighth Safety-critical Systems Symposium*. Citeseer. 2000, pp. 8–10.

[52] Refsdal, A., Solhaug, B., and Stølen, K. “Risk Analysis”. In: *Cyber-Risk Management*. doi: 10.1007/978-3-319-23570-7_8. Springer International Publishing, 2015.

[53] Rehman, H. U., Asif, M., and Ahmad, M. “Future applications and research challenges of IoT”. In: *2017 International Conference on Information and Communication Technologies (ICICT)*. doi: 10.1109/ICICT.2017.8320166. IEEE. 2017, pp. 68–74.

[54] Sklyar, V. and Kharchenko, V. “Assurance case driven design for Internet of Things”. In: *WSEAS Transactions on Computer Research* vol. 4 (2016), pp. 173–182.

[55] Spriggs, J. *GSN-The Goal Structuring Notation: A Structured Approach to Presenting Arguments*. doi: 10.1007/978-1-4471-2312-5. Springer Science & Business Media, 2012.

[56] Taubenberger, S. et al. “Problem analysis of traditional IT-security risk assessment methods–An experience report from the insurance and auditing domain”. In: *IFIP International Information Security Conference*. doi: 10.1007/978-3-642-21424-0_21. Springer. 2011, pp. 259–270.

[57] Toulmin, S. E. *The uses of argument*. doi: 10.1017/CBO9780511840005. Cambridge university press, 2003.

[58] Trujano, F. et al. “Security analysis of dji phantom 3 standard”. In: *Massachusetts Institute of Technology* (2016).

[59] Voas, J. and Laplante, P. A. “IoT’s certification quagmire”. In: *Computer* vol. 51, no. 4 (2018). doi: 10.1109/MC.2018.2141036, pp. 86–89.

[60] Wang, R. et al. “Safety case confidence propagation based on Dempster–Shafer theory”. In: *International Journal of Approximate Reasoning* vol. 107 (2019). doi: 10.1016/j.ijar.2019.02.002, pp. 46–64.

[61] Zhou, B. et al. “Smart home energy management systems: Concept, configurations, and scheduling strategies”. In: *Renewable and Sustainable Energy Reviews* vol. 61 (2016). doi: 10.1016/j.rser.2016.03.047, pp. 30–40.

[62] Zillner, T. and Strobl, S. “ZigBee exploited: The good the bad and the ugly”. In: *Black Hat–2015 https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly-wp.pdf* (2015).

Part II

Scientific Contributions

Paper I

A Methodology for Security Classification applied to Smart Grid Infrastructures

Manish Shrestha, Christian Johansen, Josef Noll, Davide Roverso

Published by Elsevier in the International Journal of Critical Infrastructure Protection, Volume 28, March 2020.

DOI: 10.1016/j.ijcip.2020.100342

A technical report supporting this publication was published in December 2017, with research report number 476, ISBN 978-82-7368-441-7, and ISSN 0806-3036, which is superseded by this journal version and thus not included in this thesis.

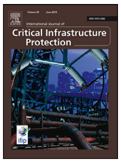
The report can be found at:

http://urn.nb.no/URN:NBN:no-63576

Contents lists available at ScienceDirect

International Journal of Critical Infrastructure Protection

journal homepage: www.elsevier.com/locate/ijcip



A Methodology for Security Classification applied to Smart Grid Infrastructures

Manish Shrestha[a,b], Christian Johansen[a,*], Josef Noll[a], Davide Roverso[b]

[a] *Department of Technology Systems, University of Oslo*
[b] *eSmart Systems, Halden, Norway*

A R T I C L E I N F O

Article history:
Received 27 August 2018
Revised 24 October 2019
Accepted 11 February 2020
Available online 17 February 2020

Keywords:
Cybersecurity
Security classification
Smart grid
Certification
Advanced metering infrastructures

A B S T R A C T

The electricity grid is an important critical infrastructure that is undergoing major changes, due to the Internet of Things (IoT) and renewable energy, heading towards the smart grid. However, besides the many good promises of the smart grid, such as better peak control, cheaper maintenance, and more open energy markets, there are many new security threats evolving, especially from the IoT side, and also from the diversification of the systems and practices that the smart grid brings. We thus see the need for more light-weight and dynamic methods for conducting security analyses of systems applicable at (re)design time, intended to help system engineers build secure systems from the start. As a consequence, the methods should also look more at the functionalities (exposure/protection) of the system than at the possible attacks.

In this paper we propose a methodology called Smart Grid Security Classification (SGSC) developed for complex systems like the smart grid, focusing on the specifics of Advanced Metering Infrastructure (AMI) systems. Our methodology is built upon the Agence nationale de la sécurité des systémes d'information (ANSSI) standard methodology for security classification of general Information and Communication Systems (ICS). Analyses performed following our method easily translate into ANSSI valid reports. Our SGSC is related to methods of risk analysis with the difference that our classification method has the purpose to assign a system to a security class, based on (combinations of) scores given to the various exposure aspects of the system and the respective protection mechanisms implemented; without looking at attackers. There are multiple uses of SGSC, such as offering indications to decision-makers about the security aspects of a system and for deciding purchasing strategies, for regulatory bodies to certify various complex infrastructure systems, but also for system/security designers to make easier choices of correct functionalities that would allow to reach a desired level of security. Particularly useful for smart grid systems is the discussion and mapping that we do of the SGSC methodology to a complex AMI infrastructure description derived from real deployments being done in ongoing Norwegian smart grid upgrades.

1. Introduction

With the increase of population and advancement of technology, the demand for energy is increasing. However, the environmental problems demand more efficient use of resources. The inclusion of renewable energy sources like solar and wind are often unreliable compared to traditional sources like coal or nuclear power. Moreover, these are difficult to integrate with the traditional electricity grid, which is already hard to manage and monitor. The smart grid has the potential to solve such problems by embracing the Internet of Things (IoT) paradigm.

The development of IoTs is driving technology and computing ability into the smallest devices, making data collection and availability ubiquitous. However, IoT expose more devices to cyberattacks, which are more threatening than ever before. On average, it takes two minutes for an IoTs device to be successfully attacked[1]. According to Symantec's Internet Security Threat Report from 2017, on average there were nine attacks every hour against Symantec honeypots.

* Corresponding author.
E-mail addresses: manish.shrestha@esmartsystems.com (M. Shrestha), cristi@ifi.uio.no, christian@johansenresearch.info, cristi@angeloti.info (C. Johansen), josef@jnoll.net (J. Noll), Davide.Roverso@esmartsystems.com (D. Roverso).

[1] Symantec Corporation, "Internet Security Threat Report (ISTR), Volume 22". https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf.

https://doi.org/10.1016/j.ijcip.2020.100342

Smart grid forms one of the largest network as it shall connect each and every end node (houses, industries, buildings etc.) that have access to electricity. Along with increasing connectivity, it collects massive amounts of data which may contain critical and private information. Protecting such complex infrastructures from cyber-physical attacks is a big challenge.

The electricity grid in Norway is already in transition towards a smart grid, e.g., several Distribution Systems Operators (DSOs) have already started to install smart meters to replace traditional electricity meters. By 2019, all electricity consumers in Norway are required to have smart meters[2]. A smart metering infrastructure invites the opportunity for several new attack surfaces that do not exist in traditional metering.

The work of Hansen et al. [1] studied Advanced Metering Infrastructures (AMI) deployments in the United States, describing AMI infrastructure components and attack surfaces. These authors identified potential attacks and impacts of attacks on AMI, which can be then used in risk assessment. Their work concludes by describing the lack of a unified framework for analysing security aspects of complex smart grid systems.

Therefore, in this paper, we propose a methodology for security classification of complex systems relevant for smart grids. Particularly, we apply this methodology to an AMI infrastructure derived from deployments done by Norwegian DSOs using Kamstrup equipment. This real-life deployment demonstrates that the security classification methodology proposed here is well suited for AMIs and transitively for other smart grid systems.

The proposed security classification methodology is based on the ANSSI standard developed for general ICSes. Additionally, we follow other standards such as the relevant European Union Agency for Network and Information Security (ENISA) recommendations for smart grids. We also position our work with respect to relevant risk analysis and classification work related to security. In particular, we do not want to reinvent, but only adapt and enrich the existing methodologies. Our proposed methodology fits the real systems that we encounter in the smart grid deployments, at least in Norway. As far as we know, this is the first security classification methodology tailored to AMI and smart girds.

The contribution of the present work is two-fold.

Firstly, we propose a security classification methodology that is focused on the functionality of systems, i.e., on their exposure aspects and protection mechanisms. This deviates from many traditional methods where the attacker is an important part of the security evaluation, like risk-based methods (e.g., [2,3]). We are motivated by offering the system/security engineers/designers a more concrete method to guide their decisions regarding security when engineering critical infrastructure systems. To this end, the classes of our classification method will provide "goals" to be reached by the engineer's designs, whereas our method of evaluating a system into a class is meant to provide guidelines for how to implement a system to meet the desired security (hence the protection mechanisms and focus on exposures). Note that traditional methods can be applied on top of our method if and when more value-driven evaluations are needed (thus attacker models are factored in). However, for critical infrastructures, we encourage the security to be a priority and designed for from the start, thus guided by our methodology.

Secondly, our methodology is more detailed than the one from ANSSI, as it is tailored to smart grid systems, particularly looking at AMIs. For this reason, we take the effort of detailing a real AMI infrastructure, as deployed in Norway, and use these details when making various design decisions for our SGSC method (hence this name). The SGSC methodology is timely as various smart grid systems are being designed and deployed all over the world; and following our methodology would force security-by-design, thus not falling prey to the IoT systems design where security is largely missing.

Summary of the Paper.

- In Section 2 we synthesise the complete architecture of a concrete AMI infrastructure that is currently deployed in the Norwegian smart grid. The details include as much as the confidentiality concerns of the utility company allows. We use this model and detailed information as our use case of a realistic smart grid system.
- We propose a new Security Classification methodology in Section 4, that extends the ANSSI standard (presented in Section 3) while complying with the ENISA relevant documentation. The particular aspects of the proposed security classification are driven by the specifics of the smart grid systems.
- To validate the proposed smart grid security classification, we apply it in Section 5 to the AMI architecture that we detailed in Section 2.
- There are multiple motivations for having such a SGSC methodology which we mention in Section 6. The areas of relevance to our work are presented in Section 7.
- Section 8 presents conclusions and further work, which would include an implementation of our method into a tool for evaluators as well as metrics and respective automated measuring tools.

2. Smart grid and AMI

This section provides details of the actual AMI infrastructure currently being deployed in Norway. We have used these details when designing the methodology from Section 4 and we use it again in Section 5 to exemplify the applicability of our method on this realistic AMI system.

The smart grid, regarded as the next generation power grid, is envisaged to completely change the way we use and manage the electric grid today[3]. The smart grid uses a *two-way flow of both electricity as well as of information* to create, at least conceptually, a widely distributed automated energy delivery network [4]. Opposed to the traditional one-way flow from the power-plant to the consumer, the smart grid assumes the users also generate electricity [5,6]. These end users (called prosumers) can feed the generated electricity back into the grid and trade it in local or global energy markets[4].

Blackouts and accidents due to imbalance between the demand and supply of the energy are common[5,6]. The introduction of a *two-way flow of information* is meant to make the smart grid more efficient and reliable through, e.g., demand response programs [7] or use of Smart Home Energy Management Systems (SHEMS) [8,9]. There would also be an efficient use of electricity through peak shaving techniques [10].

The grid components generate large quantities of data, which can be used to create novel services using Artificial Intelligence (AI) and data analysis techniques. Deep learning methods were used in [12] to perform grid inspection to identify faults and damages in equipment on the power lines. Currently, such inspection is

[2] NVE, "Smart metering (AMS)". https://www.nve.no/energy-market-and-regulation/retail-market/smart-metering-ams/.

[3] https://ec.europa.eu/research/energy/pdf/smartgrids_en.pdf.

[4] EMPOWER project "Local Electricity retail Markets for Prosumer smart grid pOWER services" https://ec.europa.eu/inea/en/horizon-2020/projects/h2020-energy/grids/empower.

[5] http://reneweconomy.com.au/tesla-big-battery-outsmarts-lumbering-coal-units-after-loy-yang-trips-70003/.

[6] https://www.macrobusiness.com.au/2018/02/population-ponzi-overrun-victorian-power-supplies/.

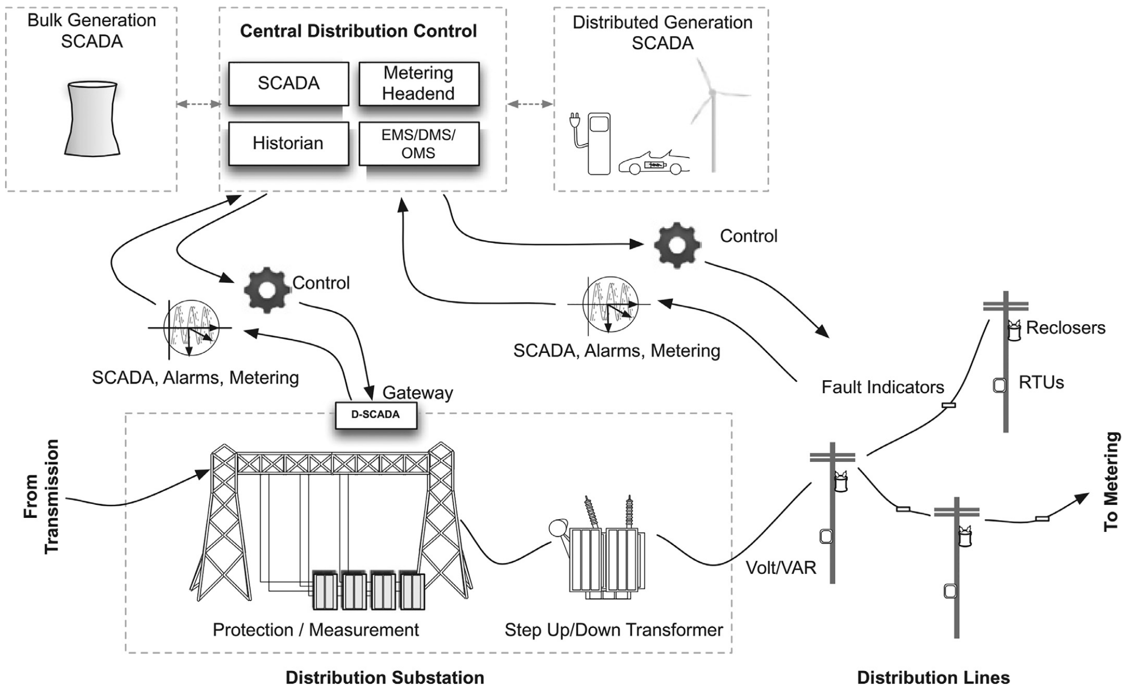


Fig. 1. Distribution Architecture; reproduced from [11, pg.44].

highly manual, slow and expensive as it is done using helicopters or through foot patrols. Similarly, Dang-Ha et al. [13] used the consumption data from smart meters to analyse consumption patterns and cluster consumers into different groups so that the retailers can provide customized services to them.

The Smart grid can be seen as the traditional electrical grid extended with IoT capabilities for information collection, monitoring and control of its equipment [14,15]. The electricity grid is typically divided into three sections: Generation, Transmission, and Distribution. In Norway, the transmission grid section has high voltages, typically 300 or 420 kV to reduce power losses while transmitting on long distances. In the regional grid, the voltage is stepped down to 132–11 kV, which is further stepped down to 400–230 V in substations before reaching industrial, commercial and residential consumers. Figure 2 shows the schematic version of the electric grid organization in Norway. The arrow in the figure represents the direction of flow of electricity, **"G"** represents generators and **"L"** represents loads like consumers, industries, etc.

We focus on the distribution part of the smart grid, and in particular on the AMI infrastructure, and therefore, we omit the details about the generation and transmission parts.

2.1. Distribution grid infrastructure

The distribution grid begins after high voltages from the transmission grid are lowered to standard distribution level voltages by step-down transformers. This part of the grid is responsible for delivering energy to the end users. Several substations are deployed and have monitoring, protection and control capabilities. The distribution grid includes several important components s.a.: Supervision Control and Data Acquisition (SCADA), Master Terminal Unit, Distribution Management System (DMS), Operation Management System (OMS) and several back-office systems at data centres or central control facilities. As shown in Figure 1, the Central Distribution Control is the main control centre of the Distribution Network which communicates with other components including distribution substations, distribution lines, and Generation SCADAs.

The SCADAs systems in the distribution network have the responsibility to control distribution operations, which include manual and automated control of load management and Dynamic Feeder Reconfiguration (DFR). DFRs dynamically collects data from distribution feeders and when a fault is detected, it isolates the fault and restores the electricity using available capacity from adjacent feeders. The SCADAs system located at a substation communicates with SCADAs located in a central control centre. SCADAs provides various functionalities including communication capability to the substation devices, data aggregation and collection, automation capability through Programmable Logic Controller (PLC), metering functionalities, fault recording and alerting, and transformer monitoring and control. Similarly, field devices consist of Remote Terminal Units (RTU), Intelligent Electronic Device (IED) or other distributed controllers.

2.2. AMI infrastructure

Advanced Metering Infrastructures (AMIs) refers to the integrated metering system with communication capabilities. It can automatically perform the measurement, collection, delivery, analysis, and storage of metering values. AMIs typically consists of smart meters, concentrators, head-end systems, and Meter Data Management (MDM) systems, along with two-way communication channels between these components. Meter values are reported at regular intervals. In Norway, the meter-values are usually collected once every hour (sometimes every 15 or 30 minutes) by the concentrators. The collected values are then reported to the head-end system via mobile network or wired network. Similarly,

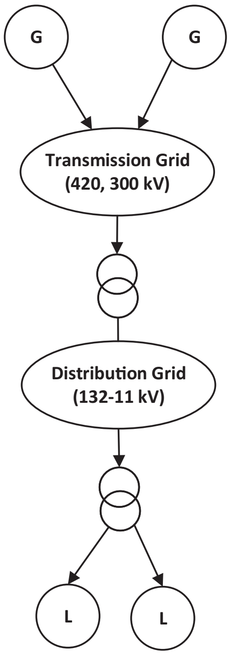

G
G
Transmission Grid (420, 300 kV)
Distribution Grid (132-11 kV)
L
L

Fig. 2. Electric Grid Structure in Norway; reproduced from [16].

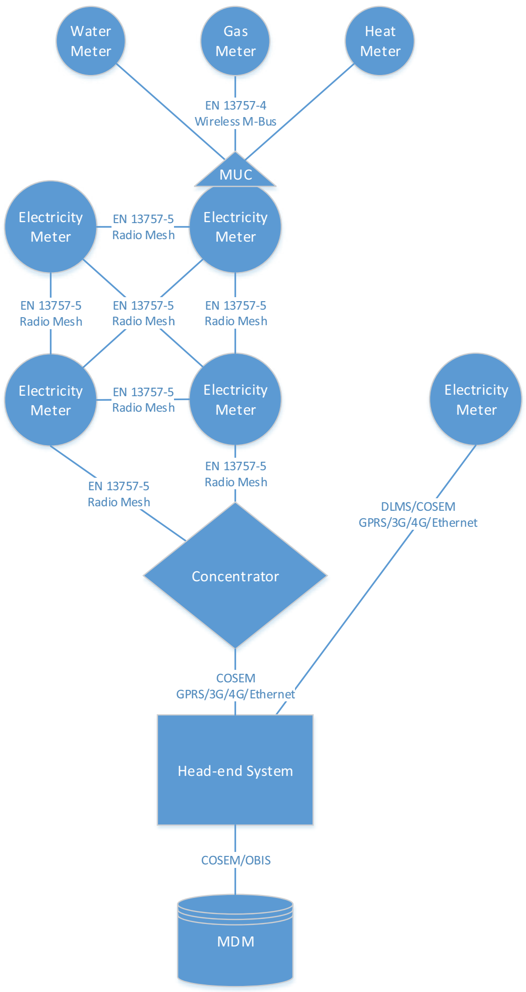


Fig. 3. OMNIA network from Kamstrup.

the command and control requests are sent to meters through the head-end systems using the same communication infrastructure. Figure 3 shows the overview of the AMIs infrastructure from Kamstrup called OMNIA Network, which is deployed widely in Norway and is our practical use case in Section 5. The major components of AMI are described below. Some details, such as key distribution hierarchies, are not disclosed here due to confidentiality requirements.

2.2.1. *Smart meters*

Smart meters are digital meters that support real-time data collection. These meters consist of a microprocessor with local storage and a communication network interface [17]. Smart meters are typically installed inside the consumer's facility (home, office, industrial) and connected to the communication network to measure electricity consumption and report consumption values or meter readings to the concentrators at regular intervals. The normal frequency of reading meter values are hourly, but it can be configured to a higher frequency, subject to legislative approval. The frequency of the readings is data sensitive, since private information can be extracted [18–20], (e.g., type of appliances, behaviour patterns, and even TV channels watched). Smart meters also have storage capability to provide historical data. Meters can also report on demand other values like events and logs such as Reactive Energy, Voltage Quality, and meter configuration [21][7]. If a meter is not able to report the values at the required time due to connection problems, it will report when the connection is re-established.

A Multi-Utility Controller (MUC) is capable of establishing interoperability between different types of meters, e.g., electricity, heat, water, providing a common point of data collection from a variety of meters. A MUC module can either be mounted on the smart electricity meter, or be standalone which communicates directly with the head-end systems[8,9].

Smart meters report energy values to the concentrator via the radio network. They also posses remote control functionalities through breaker switches that can be triggered remotely via head-end systems. It is also possible for the meters to report meter values directly to the head-end systems (without having a concentrator) through fibre optic cables, especially when the meters are located under the ground or where concentrators are not available. If a meter cannot communicate with a concentrator in a single hop, another smart meter can serve as a relay for forwarding the meter values (i.e., in a mesh-network).

[7] Kamstrup AS, "OMNIPOWER RESIDENTIAL SMART METER". http://products.kamstrup.com/ajax/downloadFile.php?uid=526905f09f7a3. Accessed: August 14, 2017.

[8] Kamstrup, "The MUC – a simple solution to a complicated task". http://products.kamstrup.com/ajax/downloadFile.php?uid=548973310558a.

[9] https://www.kamstrup.com/en-en/products-solutions/smart-grid/communication/multi-utility-data-collection.

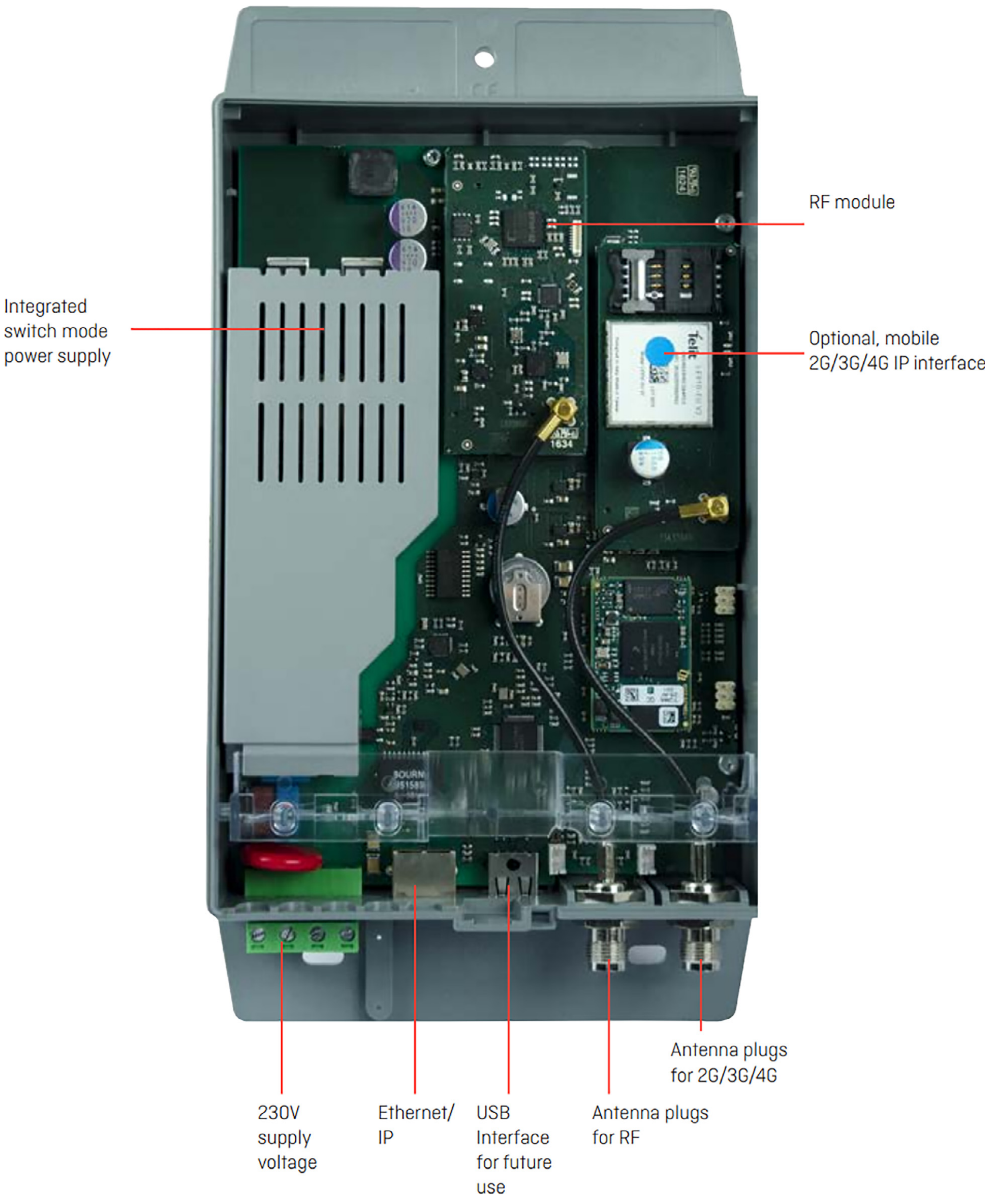

RF module
Integrated switch mode power supply
Optional, mobile 2G/3G/4G IP interface
Antenna plugs for 2G/3G/4G
230V supply voltage
Ethernet/ IP
USB Interface for future use
Antenna plugs for RF

Fig. 4. A Smart Grid Concentrator.

2.2.2. Concentrator

The concentrator acts as a mediator between the head-end system and the meters collecting data reported from the meters and reporting it to the head-end systems. When a head-end system sends out requests to receive data from meters, it is first received by the concentrator which then forwards it to the appropriate meter. The requests might be for current consumption values, events log, or open/close commands to the breaker switch of the meter. Concentrators push the values to the head-end systems whenever the values are ready. Usually, concentrators are configured to report the meter values every six hours. However, the alarms are reported in real-time. Together, smart meters and concentrators form a Neighborhood Area Network (NAN). A concentrator may send data to the head-end system using wired or wireless methods. Wired methods may include Power Line Communication, fibre optic or DSL (Digital Subscriber Line) [22]. Communication infrastructure using fibre optic and DSL are expensive. Though Power Line Communication is cheaper than fibre optic and DSL, it has low data rate and is prone to signal losses and channel interference. Cellular network is often used as a preferred wireless method, due to its low installation cost. Figure 4 shows a typical concentrator and its components. A concentrator may also be capable of upgrading firmware for itself and its meters. It may contain its own local web server allowing configuration and other services through an ordinary web browser. In case Wide Area Network (WAN) connections are unavailable, the concentrator may store the values until the connection is restored and flush the stored values after they have been delivered successfully to the head-end systems.[10]

2.2.3. Head-end system

The head-end system is the central part of the AMIs. Head-end systems collect all the data from the meters and also provide the interface for the MDMs systems (see Section 2.2.4). Head-end systems expose the meter functionality to other IT (Information Technology) systems via web-services and often hide the metering complexity from the upstream IT systems. Typically, the head-end services are exposed via Representational State Transfers (RESTs)ful web APIs and COmpanion Specification for Energy Metering (COSEM) objects. All data items exposed by a meter are uniquely identified by an OBject Identification System (OBIS) code. COSEMs is an interface model to communicate with energy metering equipment, providing a view of the functionality available through the communication interfaces using an object-oriented approach [23–25][11]. The head-end system is also capable of sending commands to the meters via concentrators. Sometimes, the meters are set up to communicate with head-end systems without concentrators. In such cases, the commands are sent to the meters directly. A head-end system stores data for a limited period of time, typically for a couple of days. The head-end is a critical system and is located securely in the DSO's perimeter.

2.2.4. Meter Data Management (MDM) systems

MDMs systems are integrated with head-end systems to collect data from meters on regular intervals. The major task of MDM system is to perform Validation, Editing and Estimation (VEE) of AMI data. MDMs also offer long term storage of meter reading data as information that can be reported to other utility applications including billing, customer information systems, and outage management systems. MDM is a key resource for managing large quantities of meter data[12].

2.3. Communication channels

In this section, we discuss major WSN (Wireless Sensor Network) standards used in smart grid systems and particularly in the Kamstrup AMI infrastructure that we investigate. We also mention WANs style communication channels.

2.3.1. EN 13757

The European standard EN 13757-4 specifies the communication between utility meters and concentrators, and was developed as the standard for remote reading of utility meters in Europe. Utility meters usually include water meters, heat meters, gas meters, and electricity meters. EN 13757-5 specifies the wireless relaying protocol forming the radio mesh network of meters. Wireless M-Bus, which is widely used in metering communication in Europe, complies with the EN 13757 standard. There are several modes specified at various frequencies as shown in the Table 1[13].

Table 1
Wireless M-Bus modes.

Mode	Freq. (MHz)	Notes
S (Stationary)	868	Meters send data few times a day
T (Frequent Transmit)	868	Meters send several times a day
C (Compact)	868	Higher data rate version of mode T
N (Narrowband)	169	Long range, narrow band system
R (Frequent Receive)	868	Collector reads multiple meters on different frequency channels
F (Frequent Transmit and Receive)	433	Frequent bi-directional communication

2.3.2. IEEE 802.15.4

The standard IEEE 802.15.4 is a low data-rate wireless communication standard. Other protocols like Zigbee and WirelessHART are based on this standard. Zigbee is a low-cost, low power, two way, wireless communication standard from Zigbee Alliance, which in addition to IEEE 802.15.4 defines a communication layer on layer 3 and up (i.e., Network, Transport, and Application layers). A Zigbee device can act as an end device, a router, and a coordinator at the same time. The *Zigbee Smart Energy Standard*[14] provides the specification of the smart energy devices and clusters required to build an energy-management system that has enabled Zigbee for automatic metering, demand response, and prepayment applications, which are needed by utilities. A cluster is a set of message types related to a certain device function (e.g., Door Lock, Metering, etc.). Zigbee is widely used for both Home Area Networks and Neighbouring Area Networks.

2.3.3. Cellular communications

Cellular technologies enable communications between concentrators and head-end system using WANs. Wired networks, though an expensive alternative, are in use where cellular networks are unavailable. Concentrators support 2G, 3G and, 4G technologies. Thus, if a 4G network is unavailable, concentrators may downgrade to 3G and even 2G networks to establish communications. 5G is the next generation in cellular communication expected to be suitable for IoT[15] and can also be used for AMI in the near future.

2.4. The OMNISOFT UtiliDriver®

OMNISOFT UtiliDriver[16] is a Kamstrup head-end system that offers interfaces supporting integration with VisionAir[17] (a Kamstrup MDM system) and partner MDM systems. It is responsible for handling all communication technologies and meter types supported by Kamstrup OMNIA, in order to ensure interoperability and integration. UtiliDriver is composed of three logical layers, namely: Controller Layer, Core Layer, and Service Layer. The controller layer handles communications using specific meter technologies including GPRS (General Packet Radio Service) and radio mesh network. The core layer gathers information from the controller layer, handling common jobs and acting as a temporary cache of job results.

[10] Kamstrup AS, "Data sheet: OMNICON data concentrator". http://products.kamstrup.com/ajax/downloadFile.php?uid=591ab48ec5643. Accessed: August 14, 2017.

[11] DLMS User Association, "What is COSEM?". www.dlms.com/faqanswers/generalquestions/whatiscosem.php.

[12] US Department of Energy Office of Electricity and Energy Reliability, *NETL Modern Grid Strategy for Advanced metering infrastructure* from 2008.

[13] SILICON LABS, "An Introduction to Wireless M-Bus". http://pages.silabs.com/rs/634-SLU-379/introduction-to-wireless-mbus.pdf.

[14] Zigbee Alliance, "Smart Energy Profile 2". http://www.zigbee.org/wp-content/uploads/2014/11/docs-07-5356-19-0zse-zigbee-smart-energy-profile-specification.pdf.

[15] https://www.i-scoop.eu/internet-of-things-guide/5g-iot/.

[16] https://www.kamstrup.com/en-en/products-solutions/smart-grid/system-software/utilidriver.

[17] https://www.kamstrup.com/en-en/products-solutions/smart-grid/system-software/visionair.

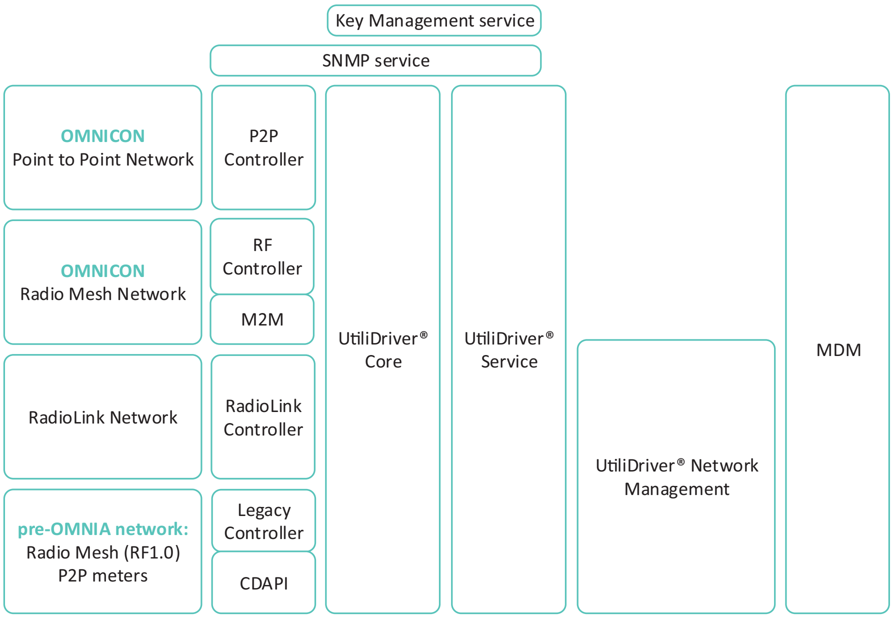


Fig. 5. Components of Utilidriver Head-end System. Simple Network Management Protocol (SNMP), Point to Point (P2P), Radio Frequency (RF), Machine to Machine (M2M), Meter Data Management (MDM).

This layer has information about the communication infrastructure and meters under its management, e.g., configuration, encryption keys, etc. Similarly, the service layer is responsible for the external interface towards the clients, which includes RESTful web services. Figure 5 shows the architecture of UtiliDriver.[18]

3. The ANSSI Security Classification as Baseline

In this section we present in some detail the *Security Classification of Complex Systems* recommendations (standard and method) made by the French Agency ANSSIs[19]. We will extend this method in Sections 4 and 5 by making it more concrete and better fit to AMI systems. Particularly, we focus on detailing the connectivity aspects in Sections 4.3 and 5.2 and the functionalities, which we call protection, in Sections 4.2 and 5.3.

The ANSSIs classification method is developed specifically for Industrial Control Systems (ICS) and has simple steps and parameters to compute the security of ICS. Figure 6 summarizes the ANSSIs classification method. A class is determined by the level of *Impact* and *Likelihood*. The likelihood is the result of combining three aspects: the *Exposure*, the users' *Accessibility* to ICSs (Users block in Fig. 6), and the level of *Attackers*. Exposure is determined by combining the *Connectivity* of ICSs and the *Functionalities* supported by the system. We further describe the classification method and its terminologies used by ANSSIs.

3.1. Connectivity

The ANSSIs classification method divides connectivity into the following five levels:

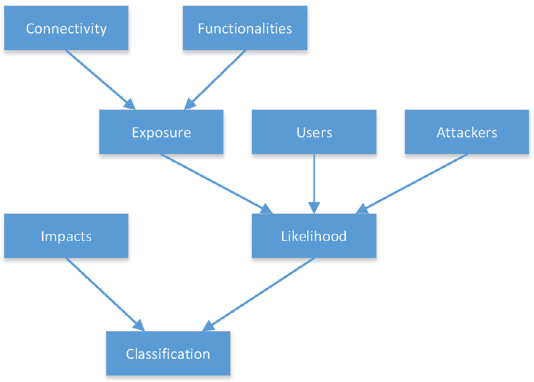


Fig. 6. ANSSIs Classification Method.

- **Connectivity 1 (C1)**: It includes completely closed and isolated ICSs.
- **Connectivity 2 (C2)**: It includes ICSs connected to a corporate Management Information System (MIS) which does not permit any operations from outside the network.
- **Connectivity 3 (C3)**: It includes all ICSss using wireless technologies.
- **Connectivity 4 (C4)**: It includes the ICSs with private infrastructure which may permit operations from outside (e.g., Virtual Private Networks (VPNs), Access Point Names (APNs), etc.). This system may use private or leased communication infrastructures which enable communications between the distributed systems at different sites (e.g., remote maintenance and management).

[18] Kamstrup AS, "Data sheet:OMNISOFT UtiliDriver 3.10 ". http://products.kamstrup.com/ajax/downloadFile.php?uid=537d9fe634eff. Accessed: August 14, 2017.
[19] ANSSI Classification Method and Key Measures from 2014.

Table 2
ANSSI's ICSs Exposure, combining Functionality and Connectivity.

F3	E3	E3	E4	E4	E5
F2	E2	E2	E3	E4	E5
F1	E1	E2	E3	E4	E5
Functionality / Connectivity	C1	C2	C3	C4	C5

- **Connectivity 5 (C5)**: It includes Distributed ICSs with public infrastructure. It is similar to the C4 category except that the communication infrastructure is public in C5.

3.2. Functionalities

Functionality is divided into three levels based on the complexity of the system:

- **Functionality 1 (Minimal Systems)**: ICSs which have components limited to sensors/actuators, remote I/O, PLCss, Human Machine Interfaces (HMIs)s, embedded systems, and analysers.
- **Functionality 2 (Complex Systems)**: ICSs which have Fun. 1 components as well as SCADAs systems, excluding programming consoles and engineering workstations.
- **Functionality 3 (Very Complex Systems)**: All other ICSs that do not fall into the Fun. 1 or 2 category.

3.3. Exposure

Exposure is a five level scale which is determined by combining functionality and connectivity as in Table 2. We extend and detail the concepts of exposure in Section 5.2 to fit with specific aspects of smart grids and AMIs.

3.4. Users and their authorization level

ANSSIs defines the accessibility of an ICSs based on the authorization level of the users, and identifies four categories:

- **Users 1 (authorized, certified and controlled)**: users within this category are explicitly authorized to intervene, have relevant certified expertise, and their actions are being tracked, logged, and accountable.
- **Users 2 (authorized and certified)**: this category represents all authorized and certified users where one or more possible operations are not tracked.
- **Users 3 (authorized)**: this category represents all authorized users who have no special requirements.
- **Users 4 (unauthorized)**: this category represents all the ICSs with the possibility of unauthorized intervention.

3.5. Attackers

ANSSIs have categorized the attacker's level (often called power of the attacker) into five levels[19]. It is often desired to quantify this power, e.g., in terms of money and time resources, to supplement the simplified levels given in Table 3.

3.6. Likelihood

The likelihood of an attack is estimated from exposure of the system components, users' accessibility to the systems and the attacker level. ANSSIs defines four likelihoods: (1) Very Low, (2) Low, (3) Moderate, (4) Strong. Likelihood can be calculated as follows:

$$L = E + \left\lceil \frac{A + U - 2}{2} \right\rceil$$

Table 3
Attacker's Level Scale according to ANSSI.

Attacker level	Designation	Description / Examples
1	Non-targeted	Virus, bots, etc., that are not targeted to any particular person or organization.
2	Hobbyist	Individuals with very limited means, not necessarily intending to cause harm.
3	Isolated Attacker	Individual or organization with limited means, but with a certain determination (e.g., terminated employee).
4	Private Organization	Org. with substantial means (e.g., terrorism, unfair business practices).
5	State Organization	Organization with unlimited means and very strong determination.

where: L = Likelihood; E = Exposure; A = Level of the attacker; U = Accessibility of the user; and the mathematical operator $\lceil . \rceil$ denotes rounded up value to an integer.

3.7. Impacts

The *impact* measures the consequences of an attacker compromising a system, calculating the impacts on business, government, or society sectors, which could affect physical infrastructures, human life, environment, economy, etc. The *Impact* is divided into five levels: (1) Insignificant, (2) Minor, (3) Moderate, (4) Major, (5) Catastrophic. We extend and detail in Section 5.1 with impacts specific to AMI.

3.8. Classification

ANSSIs divides ICSs into three classes and provides guidance on how to identify the class of a given system. Each class has different levels of requirements; essentially, a higher class talks about higher impacts and stronger security requirements. Below is a short description of each class defined by ANSSIs.

- **Class 1**: The ICSs in this class have low risk or impact of an attack and security measures can be applied internally. However, systems in this class need the implementation of the chain of responsibility for cybersecurity. Class 1 also requires risk analysis for cybersecurity. Physical, logical and application inventories of the ICS should be prepared. In this class, internal audit is acceptable.
- **Class 2**: The ICS in this class have a significant impact in case of an attack. Security assessment should be done by responsible third parties, and the evidence of implementation of required measures must be provided. A chain of responsibility for cybersecurity needs to be implemented. This class also requires risk analysis using a proper method done by a responsible entity. In this class, the inventories should be reviewed regularly as defined by the responsible entity and each time the ICS is modified. Internal audit is not accepted; instead, it should be carried out by external service providers.
- **Class 3**: The ICS in this class have a critical impact in case of an attack. Therefore, the highest level of measures should be taken and be verified by the state authority or an accredited body. In addition to the implementation of a chain of responsibility for cybersecurity, the identity and contact details of the person in charge of this chain should be communicated to the cyberdefense authority. This class requires a detailed risk analysis done by a certified service provider, and which should be reviewed regularly at least every year. The inventories should

be reviewed at least once per year. Audit of the ICSs in this category must be carried out by an independent, certified service provider.

In conclusion, the ANSSI classes focus on how security measures should be applied to ICS systems, and reflects the worst severity and the likelihood of attacks on the system. ANSSI provides a separate document called detailed measures which provide the guidelines on taking measures for each type of class. However, at design-time we do not need to consider the attacker, so to be able to design for future worst cases, thus promoting strong security by design. Attacker models and risk analyses could still be applicable in a second stage where trade-off evaluations between security costs and loses can be made and decisions taken, but this can be independent of the security classification.

More importantly, our goal is to provide a more concrete classification of the connectivity and protection mechanisms, useful for system designers to build their systems to reach their desired class. This is what we do in the next section where we introduce a compatible extension of ANSSI.

4. Smart Grid Security Classification

A traditional process of risk assessment is based on *impact* and *likelihood* of threats (see more in the Related Work Section 7). A common method of assessing the security of a system is to find all the assets under the influence of the system and to identify the attack approaches by which assets might be compromised. Based on this, a risk assessment is done, and all the risks under considered attacks are identified. These are attack-centric approaches for risk assessment, and the whole process must run each time the risk assessment is performed. However, there is a constant increase in the number of attacks on IoT[20] as well as an increase in the availability of sophisticated attack tools; which means that new attacks need to be taken into consideration and assessment should be performed more often. Traditional risk assessment methods have difficulty keeping up with emerging threats and using dynamic and automated approaches to security [26]. We thus see attack-centric risk assessment methods as incurring potentially high costs for *developing secure IoT systems* when applied on a final system and reiterated on each (quite often) update of that system [27].

We often see that IoT systems usually have certain commonalities regarding their composition, communication, interaction, attack surfaces, consequences, etc. It is thus useful to identify general characteristics of IoT systems and group them so that every new system under investigation (and its components) falls into one of the groups. This offers the advantage of having a set of common features, making it easier to identify necessary risks and precautions that need to be taken to secure a new system. This information is helpful to both security professionals and management officials to get a quick overview of the criticality of the systems they are dealing with.

We thus propose the notion of *Security Classes* to enable analysts to focus on grouping complex systems into predefined classes with respect to security requirements. The particular details that we consider are guided by our primary intended application to smart grids, thus we here call the method and respective classes *Smart Grid Security Classification/Classes (SGSC)*. Our SGSC methodology is general enough to be adapted to other complex systems (e.g.,in [9] we apply it to SHEMS). Since we extend the ANSSIs standard, then to any system where our SGSC method can be applied, the ANSSIs method should be applicable as well. In addition to the results obtained from SGSC, ANSSI classification requires likelihood to be determined, which involves accessibility, power of attacker, and functionality levels (see Section 3.6). Details about the system, like connectivity, functionality and security mechanisms, are already known from the SGSC. Therefore, one can easily extract an ANSSIs compliant version from a SGSC evaluation of a system (with some details being omitted).

[20] Symantec Corporation, “Internet Security Threat Report, Volume 23”. www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf.

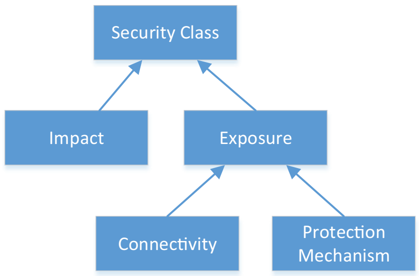


Fig. 7. Basic inputs for defining a security class in our SGSC.

The SGSC method analyses two aspects: how important a given (sub)system is (equivalent to the Impact in ANSSIs, also called criticality of this subsystem), and what functionality surface it provides to be attacked, which we call *exposure of the system*. Thus, an SGSC class is the result of combining the consequence of an attack on the given (sub)system and the exposure of the (sub)system to the attacker, as sketched in Figure 7. The ANSSIs notions of *user* and *attacker* are incorporated, in our case, under the calculation of exposure.

4.1. Consequences of an attack

We adopt the five levels for characterising an attack from the ANSSIs standard (see Subsection 3.7). Classifying the impact into one of the five levels is highly specific for the system under evaluation, the types of impacts it may have (i.e., financial, social, etc.), or the area of application. Therefore, usually, methods and standards leave this open and to the decision of the security/risk analysts. However, guidelines are always useful (and also found in the literature[21] [28]), but even more useful would be guidelines presented as templates with input fields, maybe some being only drop-down lists to choose from, and with a calculation software[22] behind that would aggregate the numbers given by the analysts, like the Multi-Metrics approach to measurable security of Fiaschetti et al. [29]. Then for each field or type of loss or impact, the work of the analyst reduces to a simpler task of providing a numeric scale for a particular impact factor. For example, assume a financial consequence for which the analyst could prepare a scale as follows. We can say that if the loss is above 1 billion, it is catastrophic. Similarly, if the loss is between 10 million and 1 billion, then it is Major; if the loss is between 5 million and 10 million, then it is a Moderate loss; if the loss is between 1 million and 5 million we call it Minor; and less than 1 million, we call it Insignificant. Such scales could be even openly available or shared inside a consortium, and made for various sectors and groups of losses, assigning a level from 1 to 5, taking the most extreme loss/impact as level 5.

[21] ISO/IEC 27005 Information technology – Security techniques – Information security risk management (second edition).

[22] In future we plan to focus on developing such automated calculating methods for SGSC based on the Multi-Metrics approach to measuring the security of a complex system from its components [29–31].

Table 4
Referred sources for the construction of security criteria.

Protection criteria	Source
Data Encryption	ISO 27002, OWASP, ETSI
Communication and Connectivity Protection	IIC, ISO 27002, ETSI
Software/Firmware Security	ISO 27002, OWASP, ETSI
Hardware-based Security Controls	CSA
Access Control and Authentication	ISO 27002, OWASP, IIC, CSA, ETSI
Cryptography Techniques	IIC, ISO 27002
Physical and Environmental Security	ISO 27002, OWASP, CSA
Monitoring and Analysis	ISO 27002, OWASP, IIC, CSA, ETSI

Thus, for evaluating consequences of attacks we use the same level of abstraction as ANSSIs with definition from Section 3.7. Details for impacts related to AMI are given in the application Section 5.

4.2. Security criteria

We evaluate a system's security on the basis of *security criteria* presented below, derived from guidelines provided by widely used sources such as ISO 27002, Open Web Application Security Project (OWASP), ENISAs and best practice guides for IoT from Cloud Security Alliance (CSA) and Industrial Internet Consortium (IIC)[23],[24].

The ISO 27002 standard[25] provides generic guidelines on security controls for information security management systems. This standard includes several security objectives along with possible controls to fulfil them. Similarly, OWASPs provides IoT security guidance for developers[26]. Unlike ISO, which is focused on information security of an IT environment of an organization, OWASP guidelines focus on cybersecurity of IoT systems. ENISA also provides guidelines for securing smart grid systems. They propose 10 different domains (similar to ISO security objectives) along with the controls for each domain (see Section 7).

CSA and ICC are non-profit organizations that focus on enabling technologies for secure Industrial IoT; their security guidelines for IoT are also useful for smart grid security. In Table 4 we summarize the sources for the selected security criteria relevant for our methodology.

Data encryption. Sensitive data should always be encrypted during transport and storage. When implementing such mechanisms, proprietary protocols should be avoided. It should be ensured that Secured Sockets Layers (SSLs)/Transport Layer Securitys (TLSs) implementations have proper configurations and are up to date [32].

Communication and connectivity protection. Communication channels between components can be protected by protecting information flow and endpoints. Endpoints have different capabilities and security requirements. This may include mechanisms like network data isolation, network segmentation, firewalls, unidirectional gateways, network access control, etc.[23]

Applications such as Email, Domain Name Servers (DNSs), Dynamic Host Cofiguration Protocols (DHCPs), etc, which run at the application level of the network, can be protected through proper network monitoring and analysis, configuration and management. Devices should have a minimum number of network ports enabled and all unused ports should be disabled. If the situation allows, devices should always avoid making the network ports and services available to the Internet.

Software/firmware security. The firmware software is the core of a component and must be secured against malicious updates and installation. Similarly, unauthorized modification of other software may result in security threats. Therefore, it should be ensured that software/firmware are protected against unintended and unauthorized updates and modifications. Update Servers (i.e., servers responsible for sending system/firmware updates to the system components) must be trusted and in a secure state so that no illegal software can be sent out as updates. For classical IT systems examples include the Windows Server responsible for handling updates for other computers in the corporate network, whereas for IoT infrastructures an update server could be responsible for over the air updates like the Zigbee OTA Upgrade Cluster.

Signing update files and validating on the devices before installation may protect illegal installation and updates. If possible, software and firmware should be updated as soon as vulnerabilities are discovered and fixes are available. There should also be the provision to implement scheduled updates. In addition, the update process of firmware should also take care of unwanted situations like network and power disruptions [33].

Hardware-based security controls. Hardware protection should go along with the software protection. Software weaknesses and misconfigurations are not the only sources of attacks in the IoTs world. One of the factors on which hardware security depends is the security of the micro-controller used in a given device. It also depends on whether a Trusted Platform Module (TPM) is integrated into the component, and whether and how it is used [34,35]. There are other mechanisms like using Memory Protection Units, incorporating Physically Unclonable Functions, using cryptographic modules, etc., that may contribute to hardware protection of the system [36,37].

Cryptographic techniques. There are two types of cryptography namely symmetric and asymmetric cryptography. In symmetric cryptographic techniques, the parties exchanging information share the secret key which is used for encrypting and decrypting messages. Whereas in asymmetric cryptography, one party distributes its public key to other parties who use it to encrypt messages that can only be decrypted using the private key, which is kept secret. Cryptographic techniques are basically used to ensure confidentiality. These techniques can be implemented for protecting communication and connectivity and establishing secure key management. Examples of its applications useful for IoT and smart grid are message authentication, protected key store, code signing, secure bootstrapping, secure patch management, mutual authentication.[23]

Physical and environmental security. Critical components should be protected against unauthorized physical access. This criterion evaluates how well the system is protected against physical access and environmental conditions. Depending on the context, it may include access control of physical perimeter (area, building, room) and set of equipment. In case of equipment, it may have several physical ports accessible which can be misused. Protection mechanisms like disabling unused physical ports or installing equipment with minimal physical ports, physical tamper detection, etc., fall under this criteria.

Access control. Access control refers to mechanisms for protecting assets from unauthorized components, based on the business and security requirements (cf. ISO 27001). Access control can be achieved through authentication and authorization mechanisms which validate the interacting components and their privileges against system access, and then apply a access control system, e.g., based on roles or attributes. Access control mechanisms can be used to protect web, cloud and mobile interfaces that can be used to interact with the system components. These interfaces have typ-

[23] IoT Working Group, Cloud Security Alliance (CSA), “Future-proofing the Connected World: 13 Steps to Developing Secure IoT Products”. https://downloads.cloudsecurityalliance.org/assets/research/internet-of-things/future-proofing-the-connected-world.pdf.

[24] Industrial Internet Consortium, “Industrial Internet of Things Volume G4: Security Framework”. https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB.pdf.

[25] ISO/IEC 27002:2013 Information technology – Security techniques – Code of practice for information security controls (second edition).

[26] OWASP, “IoT Security Guidance”. https://www.owasp.org/index.php/IoT_Security_Guidance.

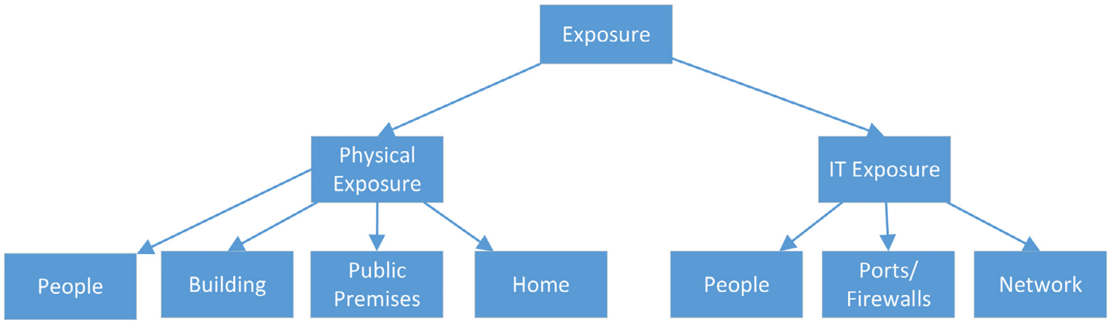


Fig. 8. Types of exposure that we consider in SGSC.

ically password-based authentication. For securing such interfaces, default passwords should not be used. Weak passwords should be avoided and if required, multi-factor authentication can be enabled to increase the strength, or even newer mechanisms s.a. [38].

Monitoring and analysis. Logging and monitoring help in tracking and analysing activities going on in the system, while also allowing for accountability.[26] In case of security incidents, monitored processes and logged data may help to understand the cause and prevent such incidents from happening again. Systems like Intrustion Detection System (IDS) and Intrusion Prevention System (IPS) help to identify and prevent attacks on the system and its components. Thus, evaluating the logging and monitoring mechanisms used by a system is an important security criterion for SGSC.

4.3. Exposure to attacks

As systems get more complex, they become harder to protect, partly because a complex system usually communicates with several external systems, and can have several interfaces available for communication. Therefore, *exposure* can be described as *the degree to which a system's interfaces are available to attacks*. We consider two types of exposures: IT Exposure and Physical Exposure (see Figure 8).

4.3.1. IT exposure

IT exposure can provide access to the system components remotely without physically being connected to the component. For instance, if a system is accessible through internet, an appropriate protection mechanism should be applied to avoid unwanted exploitation. A person can be tricked to introduce malicious codes into the network through phishing, thus people become a form of exposure [39]. This can be reduced by providing appropriate cybersecurity training to personnel. However, corrupt or disgruntled personnel may use the opportunity to harm the system as they might be authorized to access the system even after leaving the organisation [40][27]. Such exposures can be effectively reduced by implementing proper security functionalities such as access control mechanisms (like Attribute Based Access Control [41,42]), monitoring and logging.

A system can be exposed through a network which includes both wired and wireless components. If the network is isolated and physically protected, we may assume it as secure. On the other hand, if it has wireless networks, these might be accessible from far away distance using specialized antennas[28]. Furthermore, physical segregation of networks, such as control systems, are relatively rare compared with the past as systems are more connected towards the Internet and segregation is done logically rather than physically [43] [44].

4.3.2. Physical exposure

Exposure could be physical when the devices are physically available to the attacker, including any physical object or building where the device is located. It depends on how well the physical access to the system is protected from attackers. Physical exposure of certain components of a system can have a significant impact on the overall system since having physical access to one of the functioning components may allow compromising the whole system. For example, a SCADAs control room is a critical component as gaining access to it may have a critical impact, and thus, its physical protection may depend on how well is it protected against unauthorized access to the building. The protection may be put into effect by deploying security guards to protect the building or by authenticating and authorizing employees through their employee card, or even deploying biometrics authentication systems.

The physical aspect of security is comparatively less prioritized than IT security[29]. For example, it is typically assumed that the smart meter is inside the house and is accessible only by the house owner or by responsible members of the house. However, there are smart meters which are installed outside the house; e.g., for holiday cottages (see our technical report [45] for more examples). The physical security that the smart meters are equipped with usually involves putting it inside a box locked with simple locks, and often the key is hung outside (i.e., convenience over security). Moreover, many types of smart meters also come with an emergency power button (sometimes made easily reachable) which can be used to switch on/off the meter. Furthermore, even if the utility company may know about the meters that are switched off, because of safety reasons, it is not allowed to switch them on remotely. This means that an authorized person needs to physically come to the meter in order to switch it back on. Thus, even if the system (smart meter in our example) ideally belongs to some high security class, due to the environment where it is deployed, the system could be downgraded to a lower class due to the way it is deployed or configured.

Components can be located in a public environment; for instance, concentrators are typically placed in the public area

[27] https://archives.fbi.gov/archives/newark/press-releases/2011/former-shionogi-employee-arrested-charged-with-hack-attack-on-company-servers.

[28] https://www.computerworld.com/article/2968179/hackers-show-off-long-distance-wi-fi-radio-proxy-at-def-con.html.

[29] https://biztechmagazine.com/article/2016/10/why-physical-security-should-be-important-cybersecurity.

outside substation and are normally protected by locking them inside a box (e.g., [45, Fig.12]). However, the physical access to concentrators is usually limited by applying protection mechanisms through locks and tamper detection alarms which notify the utility company if the concentrator box is opened. Thus, these result in reduced exposure.

Some components may be located inside a room or a building with or without surveillance (e.g., workstations, servers). There are other components like databases and web servers that may be hosted via cloud services where it is assumed hard to physically access the machines, thus we can assume that those are isolated from physical access.

4.3.3. Connectivity

Since connectivity defined by ANSSIs fits well the smart grid case, we inherit the five levels of connectivity from Section 3.1.

4.3.4. Protection level

We divide protection into five categories of increasing cumulative sets of security functionalities meant to protect the system's components, like encryption, authentication, locks, tamper detection, etc. However, in spite of having such mechanisms, misconfigurations can introduce vulnerabilities. Thus, while evaluating the protection, the expert needs to evaluate both the security functionalities as well as how they are used and configured. The Protection Category (PC) can be correlated with the connectivity of the system.

PC 1:	Includes Physical and Environmental Protection
PC 2:	Includes PC 1 and Network Protection
PC 3:	Includes PC 2 and Wireless Protection
PC 4:	Includes PC 3 and Private Infrastructures protection
PC 5:	Includes PC 4 and Cloud protection

Each security mechanism may have different strengths, e.g., there are several authentication mechanisms, like pin code validation, user name and password, two-factor authentication, biometrics, etc. The strength of a security mechanism, for each protection category, is evaluated and ranked into different levels which we call *Protection Level (PL)*. This separation is used during the exposure evaluation when the protection category guides to identify the protection mechanisms, whereas the protection level determines the strength of the identified protection mechanisms, which are used for the explicit exposure evaluation. The evaluation of the protection level is usually done by an expert since there are too many aspects to consider for the many existing security functionalities. We thus leave out (abstract away) from our methodology further details of the PL evaluation.

4.3.5. Determining exposure

To make a system secure, appropriate security functionalities need to be applied. Different levels of connectivity require different sets of security functionality (evaluated as PL). Requirements of security functionality increase with the increasing level of connectivity so that the exposure can be maintained at the desired level. Obviously, an isolated system with no connection to the outside world needs less security functionality than one having a connection to the Internet.

Connectivity opens up the system's interface for accessibility, whereas protection limits accessibility in order to allow only legitimate agents to access the system. Therefore, the exposure level can be expressed in terms of Connectivity and Protection Level, as shown in Table 5. This table should be constructed by experts for each specific domain or class of similar systems. Exposure can be reduced by either increasing the protection level or by reducing the connectivity.

Table 5
Exposure evaluation: Connectivity (Sec. 3.1) vs. Protection Level.

PL1	E4	E4	E5	E5	E5
PL2	E3	E3	E4	E4	E4
PL3	E2	E2	E3	E3	E3
PL4	E1	E1	E2	E2	E2
PL5	E1	E1	E1	E1	E1
Protection / Connectivity	C1	C2	C3	C4	C5

Table 6
Smart Grid Security Classes.

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

4.4. Security class

We classify the quality of the security of a given system into six levels from **A** to **F**, where A represents a system with best security (highest security class). A security class is a result of combining the evaluation of *exposure* and *impact* of attacks on the system. A lower exposure level means a lower attack surface. However, if the impact is high, this may affect the security class of the system since impact is critical and lowers the security levels of the system. Table 6 describes the method of determining a security class in SGSC.

5. Mapping Security Classes onto AMIs

An AMI consists of several components as described in Section 2.2. Smart meters are distributed on a large area into the consumer's facilities, which includes homes, offices, industrial facilities, etc. In Norway, one DSO, depending on its size, can have from a few thousand to hundreds of thousands of customers[30]. Out of major smart meter vendors in Norway (Aidon, Kamstrup and Kaifa), we have chosen Kamstrup AMI for our security classification analysis, and in this section, we identify impacts and exposures of the OMNIA AMI system.

In this section we only present what aspects need to be evaluated when applying SGSC to an AMI, taking our details from the Kamstrup installations in Norway (and Section 2.2).

5.1. Impacts on infrastructure

Following the SGSC methodology from Section 4, we first evaluate the impact. Having the AMI compromised can result in unwanted consequences. We categorise the main impacts of cyberattacks on the AMI into: Economical Impacts; Infrastructure Damage Impacts; or Interruption of Services.

5.1.1. Energy fraud

If an attacker is able to modify the consumption information of meters, by simply altering the meter readings, the energy bill can be reduced or increased. An attacker can utilize physical or radio

[30] NVE, "2016 Nøkkeltall nettselskap". https://www.nve.no/Media/5534/indikator-selskap.xlsx. Accessed: August 22, 2017.

exposure to perform attacks on the *data integrity* of SM. As the concentrator also contains meter data (typically for ca. 6 h), it can also be utilized to exploit the integrity of meter data. Such actions pose economical threats, including privacy breaches for consumers, as well as service disruptions, e.g., if the meter reading is used for personalized billing services, altering meter values can have economic impacts.

5.1.2. Data theft

Data reported from smart meters have private information, both explicitly (as meta-data) as well as implicitly, e.g., by simply analyzing the consumption data, it is possible to predict the pattern of electricity usage of a house or an industrial facility. One might be able to know, e.g., if there are people at home by looking into the consumption information, thus facilitating burglary [46]. We noticed that data flow from meter and concentrators towards head-end systems, contain several other sensitive information like the network topology of meters and concentrators, radio signal strengths, events/alarms, configurations of AMI components. If the communication channel is compromised or the API service for communicating with the head-end system could be accessed, such sensitive information can be obtained. Smart meters typically have an interface called HAN interface which will allow the meter to communicate with other devices[31] which can be an attack surface.

Having control over the head-end system means having control over all the devices connected to it, including meters, concentrators and their network information. Some configuration of the meters may disable the alarms or trigger unnecessary alarms. Head-end systems consist of a Network Manager which is responsible for administration and maintenance of the communication network and is able to scan new meters or delete existing meters in a given network. The head-end system also contains a Key Management Service which is a tool that stores and manages encryption keys of the AMI components. By compromising a head-end system, and thus also the Network Manager and Key Management Service, meters can be removed from the network manager or credentials/keys associated with meters can be stolen, which might be used to insert malicious nodes into the AMI network.

5.1.3. Blackouts

Smart meters can be turned on or off remotely. This feature is typically introduced to be able to remotely disconnect the meters in case the power bill is not paid. If the API can be accessed by unauthorized entities, remote disconnect commands can be launched over a large area resulting in blackouts which would not only impact the business sector but also the safety of households. If the meter is shut down for a longer period of time, especially in winter, it can result in damage of property and even threaten life. For instance, water pipes may freeze and then crack (water expands at lower temperatures) resulting in flooding inside the house when the temperature rises in spring.

5.1.4. Increased socio-economic cost

In Norway, KILE[32] (Kvalitetsjusterte inntektsrammer ved ikke levert energi) is an expression of the total of socio-economic costs that are imposed on end-users in case of interruption of energy supply. KILE costs are paid by the grid companies as a penalty for interruption through power outages and voltage disturbances. This includes both short term (<3 min) and long term (>3 min) interruptions which must be recorded and reported according to the FASIT (Fault And Supply Interruption information Tool)[33] specification. These costs represent a quality adjustment of the grid companies' revenue frameworks and are a tool for building, operating and maintaining the network in a socio-economically optimal manner. Grid companies pay hundreds of millions NOK (Norwegian currency) as KILE cost every year[34]. The price to be paid would rise by several times if there are attacks resulting in blackouts.

5.1.5. Physical infrastructure damage

If AMI is compromised and an attacker can switch on or off several meters remotely at the same time, it can launch load altering attacks which may be able to damage the physical infrastructure of the grid [47]. Such attacks could be launched if an attacker has access to the API to send breaker control commands to the meters. Moreover, attackers may also spread malware over the AMI network to send such breaker commands. Attackers may also inject false data into the SCADA systems in order to trick it into taking malicious actions which may result in power disturbance in the electrical grid resulting in damage of physical components [48].

5.2. Exposure calculation for AMI

Following Section 4.3.5, we proceed to describe how one would calculate the exposure in the case of the AMI presented in Section 2. A smart grid is a very complex system as it consists of several systems communicating together where AMI is one of them. We consider the major components of AMI detailed in Section 2.2. The components must be exposed in some form in order to interact with each other, and at the same time, they have to be protected from unauthorized access.

5.2.1. Exposure of Head-end systems

Physical Exposure Head-end systems typically reside in the utility network domain, i.e., the servers where the head-end systems are hosted reside within the building of the utility company and typically physically protected. However, it still has a risk of insiders who might have physical access to the system. As the head-end system is connected to the corporate network, it has the risk of spreading malware via physical access like USB sticks. Head-end systems may be hosted in the cloud where the physical access to the machines/servers is limited or isolated.

IT Exposure Head-end systems exposed to the enterprise network may allow malware to be spread. Exposure is further increased if the enterprise network has wireless access points. If possible, wireless networks should be avoided in enterprise networks. There might also be unused communication ports which might be accessible and not properly protected. Services from head-end systems should rather be available via API services than through physically connected network.

API services are typically exposed by head-end systems to trusted third parties such as MDM systems to obtain meter data or send data to the meters such as commands or configurations. If these services are open and accessible through internet, they can be misused by unauthorized parties. Such services are typically protected using VPN along with lists of allowed IP addresses. However, security decreases if the VPN credentials are not well managed. Similarly, ex-employees may have access to valid credentials

[31] Norwegian regulations for HAN port: https://lovdata.no/dokument/SF/forskrift/1999-03-11-301.

[32] NVE, "Kvalitetsinsentiver KILE". https://www.nve.no/reguleringsmyndigheten-for-energi-rme-marked-og-monopol/okonomisk-regulering-av-nettselskap/reguleringsmodellen/kvalitetsinsentiver-kile/.

[33] SINTEF, "FASIT". https://www.sintef.no/globalassets/project/kile_uk/fault-and-supply-interruption-information-tool-fasit.pdf.

[34] NVE, "Avbrotsstatistikk 2016". http://publikasjoner.nve.no/rapport/2017/rapport2017_43.pdf.

after they leave their job,[35] which they might exploit to launch attacks remotely.

5.2.2. *Exposure of meters and concentrators*

Physical Exposure. AMI smart meters are located in a variety of physical environments, e.g., they can be located inside or outside a home or holiday cottage. If it is installed outside, it may be locked in a box. However, for industries, meters are located inside a building. Concentrators are typically locked in a box and installed near the substation. Meters and concentrators have physical ports (Ethernet, USB interfaces) available (see Figure 4 on page 5). However, these devices also have tamper detection mechanisms, which will notify the utilities if the case enclosing the device is opened.

If smart grid devices (smart meters, concentrators) are unattended, having access to physical ports like USB, serial ports, and Ethernet, can result in compromised devices. There might be a reset button which might not be protected and when an attacker has physical access to it, he might reset the device to factory settings with default passwords. The access to the physical ports allows for malware to be injected into the devices which would start reporting fake values to the concentrator. Also, if the concentrator is infected, it can inject malicious programs into the smart meters.

Meters may store the consumption information, activity logs or data about alarms. Stored data are not typically encrypted. By taking control over such data before being sent to the head-end system, an attacker can alter or override values. An attacker can delete data from the meter storage and prevent it from providing historical data.

IT Exposure. The concentrators typically have their own web server for configuring them. If concentrators have a default administrator password to sign in, these could be compromised. An attacker might be able to change the configuration or disable the concentrators by having access to such a web server. Therefore, it is important to avoid default passwords on all devices; e.g., the Mirai IoT bot exploited default passwords [49,50].

5.2.3. *Exposure of communication channels*

Cellular nnetworks. Head-end systems typically utilize the cellular network to communicate with concentrators. There are geographical areas which do not have good coverage, several areas still use 2G network for reporting values. 2G and 3G networks are known to have security problems and are considered insecure [51–53]. An attacker may be able to launch integrity attacks on data flowing between concentrators and head-end systems. Similarly, critical operations like firmware upgrades and updates of meters and concentrators are performed via such insecure networks. If the firmware is infected, the behaviour and outputs from the device is no longer trustworthy. An attacker might exploit insecure communication channels to eavesdrop or compromise other devices.

Radio communication between meters and concentrators. Compromising the communication channel (see Section 2.3.1) can disrupt the data flow, resulting in the violation of availability of the devices in the network. An attacker may introduce a fake node (a fake meter) and try to communicate with other nodes to join the network. If it is successful, this can then launch several attacks in the radio network [54,55]. If unsuccessful, it still can launch denial of service or denial of sleep attacks in the radio network which might prevent crucial notifications, like alarms, to be sent to the head-end system.

AMI components authenticate each other using authentication keys. These keys are managed by a Key Management Service which resides within the perimeter of the head-end system. Meters and concentrators are pre-registered in the key management server along with their keys. Only nodes with a valid authentication key are authorized to communicate. By gaining access to the encryption keys, an attacker may decrypt data flowing in the network.

5.3. *Security requirements based on criteria*

In the section above we discussed the physical and IT exposure in order to point out the attack surfaces of the system components (i.e., head-end systems, meters, concentrators, and communication channels). Here we present how one would use the security criteria from Section 4.2 as a guideline to derive security requirements for the AMI system.

One could iterate through each security criteria and analyse the system to make a list of relevant security functionalities for each system component. When considering physical and environmental security, which deals with protecting the system components from unauthorized physical access, a smart meter can be placed inside a secured area such as inside a home, or in a public area but enclosed in a box protected by a lock. In addition, a tamper protection mechanism also helps to secure such components. Moreover, disabling unused physical ports or making such ports unavailable without tampering with the device can increase security, e.g., the reset-to-factory-setting button can be made inaccessible to unauthorized persons. Similarly, transport encryption is necessary to protect the data in transit, i.e., all sensitive information must be encrypted on the network. Using TLS provides good encryption mechanisms between distributed components. In addition, having an end-to-end security mechanism is also important.

We use the security criteria of Section 4.2 to help us with extracting security mechanisms for the system component under consideration. Table 7 summarizes the correspondence between security criteria and security mechanisms identified for meters and concentrators.

6. Usability of Smart Grid Security Classes

Certifications, like Common Criteria (CC), are expensive [56], even more so for IoT [57, Sec.III], and takes a long time to obtain, e.g., [58, p.4] report the need of 45.9 person-year to achieve EAL7 for their micro-kernel of 8700 lines of C code. The Common Criteria offers an assurance continuity mechanism which includes re-evaluation and maintenance activities to handle changes in the systems.[36] Though this approach takes less time and costs than the original evaluation, these may still be a significant factor, e.g., frequent patches or enhancements with major changes in the product may lead to the invalidation of the certification of the new version of the product and therefore being difficult to maintain over time [59].

Security practices and criteria may vary among organizations within the same domain, such as AMI, which makes it difficult to attain consistency, repeatability and measurability in security. Therefore, there is a need for common, affordable and practical approaches towards security of smart grids. This idea is also supported by DIGITALEUROPE and encourages common security baselines with a common set of guidelines for security levels and requirements[37]. Another concern of DIGITALEUROPE is on IoT security labelling in which the ever-changing threat landscape provides a false sense of security to consumers, thus demanding

[35] 50% of Ex-Employees Can Still Access Corporate Apps. https://www.darkreading.com/vulnerabilities---threats/50--of-ex-employees-can-still-access-corporate-apps/d/d-id/1329672?.

[36] Common Criteria, "ASSURANCE CONTINUITY: CCRA REQUIREMENTS". https://www.commoncriteriaportal.org/files/operatingprocedures/2012-06-01.pdf.

[37] Digital Europe, "DIGITALEUROPE's views on Cybersecurity Certification and Labeling Schemes". http://www.digitaleurope.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=2365.

Table 7
Security criteria and protection mechanisms.

Criteria	Security mechanisms
Data Encryption	– data flowing between the wireless sensor nodes should be encrypted – data flowing towards the head-end systems should be encrypted – end-to-end encryption should be supported – use strong encryption algorithms – credentials should not be exposed in the network – sensitive stored data should be encrypted
Communication and Connectivity Protection	– components should be registered beforehand to be able to join the network – restrict availability of services to the Internet – use standard communication protocols – denial of service protection – disable unnecessary network ports – web servers deployed inside concentrators should not be accessible from the Internet
Software/Firmware Security	– devices should be easily updatable – updates should be authenticated – update image files should be signed before updating – update servers should be secured
Hardware-based Security Controls.	– utilize TPM or similar security chips to store keys
Access Control	– devices should be authenticated and authorized in order to join the network – default and weak passwords should not be used – disable remote access functionality if possible – API requests should be authenticated and authorized – use established access control systems s.a., based on roles or attributes d and standards like XACML
Cryptography Techniques	– message integrity should be checked using cryptographic keys – secure key management – secure bootstrapping
Physical and Environmental Security	– tamper detection protection mechanism should be implemented – unnecessary physical ports should be disabled – access to factory reset buttons should be restricted – devices should be locked in a container with authorized access
Monitoring and Analysis	– activities and events in each device should at least be logged locally – logs should be sent securely to the head-end system – analyse logs in backend systems – continuously monitor the network for unwanted activities – act on analysed data

a security labelling that is agile so to prevent this. Our approach aims to address such concerns by establishing security baselines using security criteria, aiming for dynamic certification in future through automated assessment and updated security metrics.

Our *Smart Grid Security Classification* is compatible with the established standards from ANSSI and ENISA, and therefore is a step towards harmonization of cybersecurity in smart grids. In this section, we discuss the stakeholders in smart grids to whom the notion of security classes can be valuable and in what way.

6.1. Regulatory and standardization bodies

Security classes are useful to explain a complex system in terms of the security levels of the various components, the overall connectivity, and impacts of attacks. This can be used to determine appropriate security measures. Therefore, regulatory bodies can use security classes to define security requirements for specific systems. These can be enforced through regulations and verified through a certification process that makes use of the methodology that we defined.

In Norway, NVE is the legal authority for setting up regulations related to the energy sector, and they provide guidelines on security of AMI[38]. However, it is difficult to verify whether the regulations are properly met. Security Classes can help to define parameters, like how strong security is required for specific components of smart grid systems. For example, if a Security Class of B was required for AMI, then according to table 6, there are several options that can give a class B, i.e., Exposure 2 and Major, Moderate or Minor impact, Exposure 3 and Minor impact, or Exposure 4 and Insignificant impact. It is the responsibility of Utilities to decide on how to reach class B.

6.2. Companies

Utilities can comply with the security class specified by regulatory bodies in order to maintain an appropriate security level. In order to achieve the specified level in a class, they have to break down the system into its components to evaluate the security at components level, which then will be aggregated towards the system level representing the overall security class of the system. In this way, utilities can get insights into the security aspects of the components in the system and recognize components that are preventing from reaching the desired level. Thus, SGSC helps utilities to make appropriate decisions in designing their system and selecting appropriate equipment and technology from their vendors. This is one main difference between our SGSC and certification methods which are applied after a system is built (and often deployed); SGSC is meant to be used at design time to help (using automated tools and templates) to build secure critical infrastructures like AMI.

6.3. End users

Security classes provide end users a high level overview of the security of a system. This allows to compare alternative systems, thus helping end users to make informed decisions when selecting a secure system. The details of protection mechanisms and exposure that a security class provides, can help end users (with limited technical skills) understand the level of security of their product (system) and can motivate them to avoid products with weak security features. The guidelines provided by the security class might also help end users to secure their system better, such as by adjusting configuration parameters or replacing some of the sub-components to avoid the weaker links in their system. Security classes provide customers with a sense of security assurance related to smart home systems, smart meters and other similar areas. Thus, the obligation to specify the security class would help to enable transparency towards security between the vendors and the end users, resulting in building trust relationships between the two.

7. Related work

Research on DSOs' current practices and challenges[39] shows the lack of security awareness and preparedness for cyber-attacks in the smart grid systems [60]. Almost every aspect in the smart grid has vulnerabilities, which is often the result of security risks that already exist in the general IT environment [61]. Several studies have been done to describe security issues in smart grid systems [61–65]. For example, Brunschwiler [66] has performed a thorough security analysis of the wireless M-bus and demonstrated several

[38] NVE, "Guidelines for security of AMI". https://www.nve.no/Media/5525/veiledertil-sikkerhet-i-ams.pdf.

[39] Our ongoing project IoTSec.no is a large Norwegian project focusing on "Security in IoT for Smart Grids".

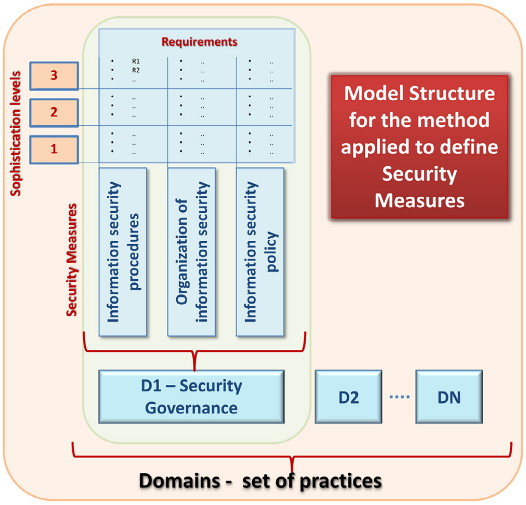

Requirements
R1
R2
Sophistication levels
3
2
1
Security Measures
Information security procedures
Organization of information security
Information security policy
Model Structure for the method applied to define Security Measures
D1 – Security Governance
D2
DN
Domains - set of practices

Fig. 9. Conceptual model of appropriate security measures.

security vulnerabilities in its implementations in smart metering infrastructures.

The ENISAs is working on standardization activities for security in smart grids. One of their documents[40] regarding certification approaches for smart grids indicates the gaps where the harmonization of security between the member states of European Union is emphasised, supporting the idea of system level security along with security and protection levels which are discussed in the ISA99 standard. Since security is dynamic, its certification should also be dynamic.

Another contribution of ENISA is on defining appropriate security measures for smart grids[41]. They have suggested ten different domains (new domains can be added) which need to be labelled with sophistication levels between one and three, representing how much the security goals have been achieved. Figure 9 describes the conceptual model for defining security measures as a matrix in which rows represent sophistication levels and the columns represent the domains. The measures/practices have been inherited from the major standards on cybersecurity, mainly ISO/IEC and NIST. To reach the targeted sophistication level, certain requirements have to be addressed and at the same time, evidence to verify the fulfilment of each requirement should be provided (mostly through documentation). ENISAs also defines a list of security measures for each domain. The ENISAs sophistication level is similar to our protection level where in our case each level specifies the adequate security functionality needed in order to reach a given level. Our methodology combines the impact of a successful attack with the protection level and connectivity in order to identify the security class of a system. Therefore, we go further than ENISAs and provide the methodology to evaluate the security measures for a security class looking at individual sub-components and then aggregating to the whole system.

[40] ENISA document nr. 9789292041014, December 2014, on *Smart grid security certification in Europe* http://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-infrastructure-and-services/smart-grids-and-smart-metering/smart-grid-security-certification/smart-grid-security-certification-in-europe.

[41] ENISA's *"Appropriate security measures for smart grids"*, Dec.2012.

7.1. Security metrics

Common Vulnerability Scoring System (CVSS)[42] is a widely adopted framework to provide the vulnerability and severity scores for a given software. The CVSSs vulnerability score is scaled between 0 and 10 where 10 is the maximum score for a most serious vulnerability. It provides three types of scores: Base Score, Temporal Score and Environmental Score. The Base Score is set by the vendors and remains the same over the time and user environment. The Temporal Score changes with time, whereas the Environmental Score depends on the users' environment. When a vulnerability is discovered, the CVSSs score helps to rank it on the basis of its severity so that the patches can be prioritized. CVSSs can be one of the attributes in our SGSC for describing the security of a system, but unpatched vulnerabilities dramatically lower the security of a system.

The work of Thomas et al. [67] proposes a security metric called FUM, which ranks the performance of device manufacturers and network operators on the basis of security updates and exposures to vulnerabilities that exist on the devices. However, the FUM score is very dynamic and needs recalculation whenever any of the devices receive updates, which might not always be feasible. Also, being updated does not always mean freedom from vulnerabilities because new unknown vulnerabilities may be introduced. This approach is limited to updates and vulnerabilities, however, in SGSC we are interested in incorporating updates and vulnerabilities together with protection mechanisms and impacts.

A "Framework for Smart Grid Cybersecurity Exposure Analysis and Evaluation" was proposed in [68] to compute exposure for a smart grid system based on access graphs. In that article, exposure is the result of computing the shortest path of an attacker to the target component. The basis of exposure calculation of Hahn and Govindarasu [68] uses subjective weights to capture the required strength in order to launch an attack. However, the work does not consider security mechanisms when evaluating the exposure. Usually, the strength of an attack depends on several factors like skills and privileges of an attacker, applied security mechanisms, cybersecurity policy of an organization, etc. This makes it difficult to model the exposure because there are many unknown variables in the model [68].

The subsequent work Hahn and Govindarasu [69] attempts to define an exposure metric and a methodology to compute it. They also evaluate usability through simulations of smart grid systems and claim that their methods can be integrated with NIST's Risk Management Framework. The methodology involves three steps: Risk Identification, Exposure Graph Development, and Exposure Evaluation. The method attempts to analyse exposure paths by determining the points that are exposed and whether they are protected/unprotected. However, the method does not allow inputs on the strength or weakness of the security mechanism. The authors claim that if the security mechanism has vulnerability, then the exposure graph needs to be reconstructed, but do not explain how to actually do that. According to their methodology, one could always get the same value. Throughout all the examples of Hahn and Govindarasu [69], the value of effort is taken as 1, which is not practical in the real world. Different security mechanisms may have different strengths, which result in different amounts of effort required for a successful attack. The effort depends also on the type and skill level of attackers. This method gives the paths through which attacks are possible if the attacker is able to bypass the security mechanism. However, it does not make any difference between weak and strong security mechanisms.

[42] FIRST Common Vulnerability Scoring System v3.0, Sep. 2017, https://www.first.org/cvss/specification-document.

7.2. Information classification

The UK Government Security Classification[43] is a guideline to be applied consistently throughout the government bodies, which defines an impact-based classification of security for information assets as three different levels: Official, Secret, and Top Secret. The *Official* class includes the information created and processed by public sectors. This class represents the components having a low threat profile. Similarly, *Secret* refers to the sensitive information that have higher threat profile. Finally, the *Top Secret* class comprises of the most sensitive information and requires the highest level of protection, having the highest threat profile. The impact of compromising such information could result in loss of several lives and could threaten the security or economic well-being of the country or allies. In our case, we look at ICT systems and subdivide them into components when calculating their security class, whereas the information assets are not in our focus. We do consider the above standard when data is involved.

Similarly, the NIST standard "FIPS PUB 199" [28] presents a generalised format for expressing the security category of an information system. The *security category* is based on impacts and does not mention the structure of the network or exposures. The security category is expressed as:

$$\mathbf{SC}(information\ system) = \{(\mathbf{confidentiality}, impact), (\mathbf{integrity}, impact), (\mathbf{availability}, impact)\}.$$

This categorisation is too general to be readily applicable in the context of complex systems as it only gives the degree of potential impact on confidentiality, integrity, and availability, without consideration to its accessibility or exposure.

7.3. Risk analysis

Categorisation of risks and how to handle them is normally done via specialised Risk Assessment methodologies or frameworks, most of them typically being based on ISO 31000 and ISO/IEC 27005 [2]. ISO 31000 provides the generic guidelines for how to conduct risk management with no specific domain or industry in target. ISO/IEC 27005 is based on ISO 31000 but it provides guidelines specifically for Information Systems. Risk assessment is a multidisciplinary process which typically has the following steps: (1) Establishment of context; (2) Risk Identification; (3) Risk Analysis; (4) Risk Evaluation; (5) Counter Measures. See Figure 10 for details.

In the risk analysis step the likelihood, impacts and other parameters are associated. In other words, it is a planned process which is followed in order to find possible breaches into a system, take into consideration the relevant ones, and devise the plan to fix them. Examples of risk assessment frameworks include CORAS [70], EBIOS[44] from ANSSI[45], TVRA[46] from ETSI[47], FAIR [71], OCTAVE [3]. Our security classification methodology involves the task of identifying subsystems and components, which is quite similar

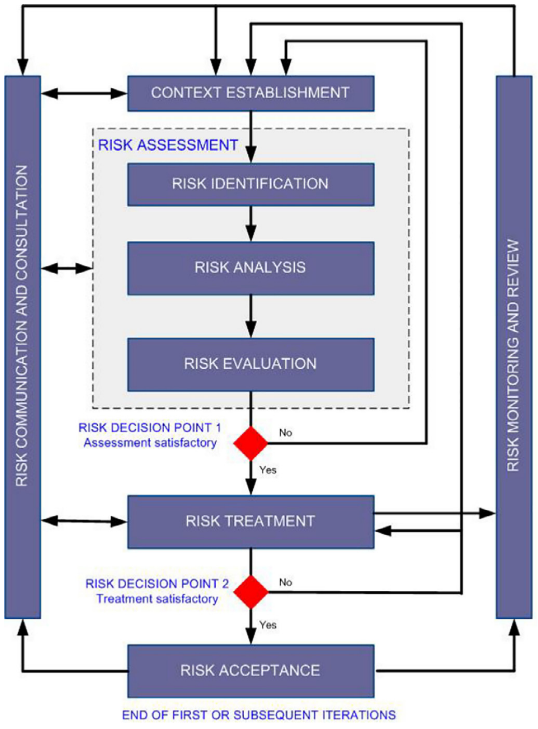


Fig. 10. ISO/IEC 27005:2011.

to context establishment. We also perform impact analysis, however, we do not involve likelihoods as a part of our methodology. We rather use exposure and impact to determine the security class, and even if an attack is less likely it still needs to be treated if the impact is significant.

7.4. Security classification for IT systems

Another attempt for Security Classification of Complex System is made by ANSSI (French Government)[19], which aims to standardise the method of classification of control systems for cybersecurity. Based on various security parameters, general guidelines are proposed for determining the security class of a control system. The ANSSI method proposes to use as basis an established Risk Analysis Method (e.g. EBIOS). The ANSSI classification method is developed specifically for Industrial Control Systems (ICS). The ANSSI method is the closest to our desires stated in the introduction. ANSSI is already seen as a standard for performing security classification for industrial control systems, and as such, our desired method for doing security classification for smart grids should conform with (maybe refine or extend) the ANSSI proposal.

We thus adopt the ANSSI standard as the basis for our work. This method has simple steps and parameters to compute the security of ICSs. However, the method of computing the *exposure* of a system does not fit the smart grid infrastructure. We extend this method to not only be based on the complexity of the system, but also on several other factors like physical access, network architecture, number of devices, open ports.

[43] UK Government Cabinet Office guidelines for *Government Security Classifications* from April 2014 https://www.gov.uk/government/publications/government-security-classifications.

[44] ENISA, "EBIOS". https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/rm-ra-methods/m_ebios.html.

[45] https://www.ssi.gouv.fr/.

[46] ETSI Technical Specification *"ETSI TS 102 165-1: Method and proforma for Threat, Vulnerability, Risk Analysis (TVRA)"*, 2017. https://www.etsi.org/deliver/etsi_ts/102100_102199/10216501/05.02.03_60/ts_10216501v050203p.pdf.

[47] The European Telecommunications Standards Institute's (ETSI) Smart Grid Technical Committee: https://www.etsi.org/technologies/internet-of-things/smart-grids and Smart Metering TC.

8. Conclusions and further work

We presented the AMI topology of Norway and discussed the security state of Advanced Metering Infrastructures based on one existing system. AMI system components are distributed in heterogeneous environments, and communicate with each other using various technologies, as detailed in Section 2.2. We have argued for the need for a security classification framework that is not attack-centric, but instead would determine the security class that a system belongs to based on measurable aspects of the subcomponents of the system and their security functionalities used to protect the various exposure aspects of the (sub)system. Our arguments have been based on our practical experience and on the related work that we presented in Section 7, ending up building our proposed SGSC methodology on the general ANSSI standard presented in Section 3. The particular usefulness of our approach has been spelt out in Section 6. Moreover, our methodology does not rule out the applicability and usefulness of existing traditional risk assessment methods; in fact, it is compatible with them, especially with the ANSSI standard.

We have thus developed a security classification method in Section 4, called SGSC, specifically focusing on details from smart grid systems. We discussed in detail in Section 5 the applicability of our method to an existing AMI infrastructure being deployed in Norway.

Tool support for future work

Even if the method presented in this paper is more detailed than the ones that we base on, we still do not achieve enough automation, and still rely on experts, like most classification methods do. It is however, our current work to provide a tool implementation of the current method which would help a non-expert in doing assessments (to some degree). We are working with Goal Structuring Notation (GSN) as implemented in the NOR-STA tool. The Goal Structuring Notation [72] is the standard maintained by the Assurance Case Working Group[48] aiming to provide a structured way to explicitly present arguments. NOR-STA is a web-based tool developed at Gdansk University of Technology [73] for logically structuring arguments complaint with OMG Argument Metamodel. The methodology used in NOR-STA is based on Toulmin's argument model [74]. By implementing our methodology in this way the one applying it would be guided on gathering justifications and evidence to support claims and argument strategies made part of the SGSC methodology. However, even with this guiding, the expert would need to provide the evidence manually, like scoring various encryption algorithms, or evaluating the importance of a sub-component or the usage of a protection mechanism.

As future work, we focus on identifying existing methods for automatically computing scores for various aspects of both exposure (like existing vulnerabilities in communication protocols and their scores) and for respective protection mechanisms. Such automated tools aim to help the security analyst by providing initial security labels/scores which then are propagated (i.e., combined) to the system level with methods like the Multi-Metrics approach of Refs. [29–31].

Acknowledgments

The authors would like to thank Yan Zhang and Elahe Fazeldehkordi for valuable inputs during several discussions. This work is funded by eSmart Systems AS and the Research Council of Norway through the projects IoTSec – "Security in IoT for Smart Grids", with number 248113/O70, and MeasurEGrid – "Measurable Security and Privacy for Services on the Smart Electricity Grid", with number 248714.

[48] GSN, "GSN Standard". http://www.goalstructuringnotation.info.

References

[1] A. Hansen, J. Staggs, S. Shenoi, Security analysis of an advanced metering infrastructure, International Journal of Critical Infrastructure Protection 18 (2017) 3–19, doi:10.1016/j.ijcip.2017.03.004.
[2] A. Refsdal, B. Solhaug, K. Stølen, Risk Analysis, Springer International Publishing, 2015, doi:10.1007/978-3-319-23570-7_8.
[3] C. Alberts, A. Dorofee, J. Stevens, C. Woody, Introduction to the OCTAVE Approach, Tech. rep, Carnegie-Mellon University, Software Engineering Institute, 2003.
[4] X. Fang, S. Misra, G. Xue, D. Yang, Smart grid – the new and improved power grid: A survey, IEEE Communications Surveys & Tutorials 14 (4) (2012) 944–980, doi:10.1109/SURV.2011.101911.00087.
[5] S.O. Ottesen, A. Tomasgard, S.E. Fleten, Prosumer bidding and scheduling in electricity markets, Energy 94 (2016) 828–843, doi:10.1016/j.energy.2015.11.047.
[6] E. Nefedov, S. Sierla, V. Vyatkin, Towards electric vehicles integration to distributed energy resources of prosumer, in: 15th International Conference on Industrial Informatics (INDIN), IEEE, 2017, pp. 769–772, doi:10.1109/INDIN.2017.8104869.
[7] P. Siano, Demand response and smart grids: A survey, Renewable and Sustainable Energy Reviews 30 (2014) 461–478, doi:10.1016/j.rser.2013.10.022.
[8] M. Beaudin, H. Zareipour, Home energy management systems: A review of modelling and complexity, Renewable and sustainable energy reviews 45 (2015) 318–335.
[9] M. Shrestha, C. Johansen, J. Noll, Criteria for security classification of smart home energy management systems, in: International Conference on Smart Information & Communication Technologies (SmartICT'19), Lecture Notes in Electrical Engineering, Springer, 2019, pp. 1–8.
[10] H. Farhangi, The path of the smart grid, IEEE power and energy magazine 8 (1) (2010) 18–28, doi:10.1109/MPE.2009.934876.
[11] E.D. Knapp, R. Samani, Applied cyber security and the smart grid: implementing security controls into the modern power infrastructure, Newnes, 2013.
[12] V.N. Nguyen, R. Jenssen, D. Roverso, Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning, International Journal of Electrical Power & Energy Systems 99 (2018) 107–120, doi:10.1016/j.ijepes.2017.12.016.
[13] T.-H. Dang-Ha, R. Olsson, H. Wang, Clustering methods for electricity consumers: An empirical study in hvaler-norway, ArXiv preprint arXiv:1703.02502.
[14] S. Goel, Y. Hong, Security challenges in smart grid implementation, in: Smart Grid Security, Springer, 2015, pp. 1–39, doi:10.1007/978-1-4471-6663-4_1.
[15] Y. Saleem, N. Crespi, M.H. Rehmani, R. Copeland, Internet of things-aided smart grid: Technologies, architectures, in: applications, prototypes, and future research directions arXiv preprint arXiv:1704.08977.
[16] S.Ø. Ottesen, Techno-economic models in smart grids: Demand side flexibility optimization for bidding and scheduling problems, Ph.d. thesis , Norwegian University of Science and Technology (NTNU), 2017.
[17] E.D. Knapp, J.T. Langill, Industrial network security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems, Syngress, 2014.
[18] U. Greveler, P. Glösekötterz, B. Justusy, D. Loehr, Multimedia content identification through smart meter power usage profiles, in: Proceedings of the International Conference on Information and Knowledge Engineering (IKE), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012, pp. 1–8.
[19] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, D. Irwin, Private memoirs of a smart meter, in: Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building, BuildSys'10, ACM, 2010, pp. 61–66, doi:10.1145/1878431.1878446.
[20] P. McDaniel, S. McLaughlin, Security and privacy challenges in the smart grid, IEEE Security & Privacy 7(3). 10.1109/MSP.2009.76.
[21] R.R. Mohassel, A. Fung, F. Mohammadi, K. Raahemifar, A survey on advanced metering infrastructure, International Journal of Electrical Power & Energy Systems 63 (2014) 473–484, doi:10.1016/j.ijepes.2014.06.025.
[22] Y. Kabalci, A survey on smart metering and smart grid communication, Renewable and Sustainable Energy Reviews 57 (2016) 302–318.
[23] S. Feuerhahn, M. Zillgith, C. Wittwer, C. Wietfeld, Comparison of the communication protocols DLMS/COSEM, SML and IEC 61850 for smart metering applications, in: IEEE International Conference on Smart Grid Communications (SmartGridComm), 2011, pp. 410–415, doi:10.1109/SmartGridComm.2011.6102357.
[24] O. Hersent, D. Boswarthick, O. Elloumi, DLMS/COSEM, John Wiley & Sons (2011), doi:10.1002/9781119958352.ch11.
[25] T. Khalifa, K. Naik, A. Nayak, A survey of communication protocols for automatic meter reading applications, IEEE Communications Surveys & Tutorials 13 (2) (2011) 168–182, doi:10.1109/SURV.2011.041110.00058.
[26] S. Taubenberger, J. Jürjens, Y. Yu, B. 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, Springer, 2011, pp. 259–270, doi:10.1007/978-3-642-21424-0_21.

[27] J.R. Nurse, S. Creese, D. De Roure, Security risk assessment in internet of things systems, IT Professional 19 (5) (2017) 20–26, doi:10.1109/MITP.2017.3680959.
[28] NIST, Standards for security categorization of federal information and information systems, FIPS PUB 199.
[29] A. Fiaschetti, J. Noll, P. Azzoni, R. Uribeetxeberria, Measurable and composable security, Privacy, and Dependability for Cyberphysical Systems: The SHIELD Methodology, CRC Press, 2017.
[30] J. Noll, I. Garitano, C. Johansen, J.D. Ser, I. Arenaza-Nuño, Perspectives in secure smart environments [29], p. 337, 2017, 337.
[31] I. Garitano, S. Fayyad, J. Noll, Multi-metrics approach for security, privacy and dependability in embedded systems, Wireless Personal Communications 81 (4) (2015) 1359–1376, doi:10.1007/s11277-015-2478-z.
[32] M. Atighetchi, N. Soule, P. Pal, J. Loyall, A. Sinclair, R. Grant, Safe configuration of tls connections, in: Communications and Network Security (CNS), 2013 IEEE Conference on, IEEE, 2013, pp. 415–422, doi:10.1109/CNS.2013.6682755.
[33] H. Mansor, K. Markantonakis, R.N. Akram, K. Mayes, Don't brick your car: Firmware confidentiality and rollback for vehicles, in: 2015 10th International Conference on Availability, Reliability and Security, 2015, pp. 139–148, doi:10.1109/ARES.2015.58.
[34] W. Arthur, D. Challener, A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security, Apress, 2015.
[35] S. Bursuc, C. Johansen, S. Xu, Automated verification of dynamic root of trust protocols, in: M. Maffei, M. Ryan (Eds.), 6th International Conference on Principles of Security and Trust (POST), Vol. 10204 of Lecture Notes in Computer Science, Springer, 2017, pp. 95–116, doi:10.1007/978-3-662-54455-6_5.
[36] R. Maes, I. Verbauwhede, Physically unclonable functions: A study on the state of the art and future research directions, in: Towards Hardware-Intrinsic Security, Springer, 2010, pp. 3–37, doi:10.1007/978-3-642-14452-3_1.
[37] T. Morris, Trusted platform module, in: Encyclopedia of cryptography and security, Springer, 2011, pp. 1332–1335.
[38] D. Migdal, C. Johansen, A. Jøsang, Offline trusted device and proxy architecture based on a new TLS switching technique, in: International Workshop on Secure Internet of Things (SIoT), IEEE, 2017, pp. 10–19, doi:10.1109/SIoT.2017.00007.
[39] H. Holm, W.R. Flores, G. Ericsson, Cyber security for a smart grid-what about phishing? in: IEEE PES ISGT Europe 2013, IEEE, 2013, pp. 1–5.
[40] J. Slay, M. Miller, Lessons learned from the maroochy water breach, in: International Conference on Critical Infrastructure Protection, Springer, 2007, pp. 73–82, doi:10.1007/978-0-387-75462-8_6.
[41] A. Lazouski, F. Martinelli, P. Mori, Usage control in computer security: A survey, Computer Science Review 4 (2) (2010) 81–99, doi:10.1016/j.cosrev.2010.02.002.
[42] X. Jin, R. Krishnan, R. Sandhu, A unified attribute-based access control model covering DAC, MAC and RBAC, in: IFIP Annual Conference on Data and Applications Security and Privacy, Springer, 2012, pp. 41–55, doi:10.1007/978-3-642-31540-4_4.
[43] C. Johnson, Securing the participation of safety-critical SCADA systems in the industrial internet of things, in: 11th International Conference on System Safety and Cyber Security (SSCS 2016), 2016.
[44] E. Hayden, M. Assante, T. Conway, An abbreviated history of automation & industrial controls systems and cybersecurity, SANS Whitepaper available at https://ics.sans.org/media/An-Abbreviated-History-of-Automation-and-ICS-Cybersecurity.pdf.
[45] M. Shrestha, C. Johansen, J. Noll, Security classification for smart grid infrastructures (long version), 2017, Tech. rep., University of Oslo.
[46] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, D. Irwin, Private memoirs of a smart meter, in: Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building, ACM, 2010, pp. 61–66, doi:10.1145/1878431.1878446.
[47] A.-H. Mohsenian-Rad, A. Leon-Garcia, Distributed internet-based load altering attacks against smart power grids, IEEE Transactions on Smart Grid 2 (4) (2011) 667–674, doi:10.1109/TSG.2011.2160297.
[48] Y. Liu, P. Ning, M.K. Reiter, False data injection attacks against state estimation in electric power grids, ACM Transactions on Information and System Security (TISSEC) 14 (1) (2011) 13, doi:10.1145/1952982.1952995.
[49] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, Ddos in the iot: Mirai and other botnets, Computer 50 (7) (2017) 80–84, doi:10.1109/MC.2017.201.
[50] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, et al., Understanding the mirai botnet, in: USENIX Security Symposium, 2017, pp. 1092–1110.
[51] A. Bais, W.T. Penzhorn, P. Palensky, Evaluation of UMTS security architecture and services, in: International Conference on Industrial Informatics, IEEE, 2006, pp. 570–575, doi:10.1109/INDIN.2006.275624.
[52] C. Xenakis, L. Merakos, Security in third generation mobile networks, Computer communications 27 (7) (2004) 638–650, doi:10.1016/j.comcom.2003.12.004.
[53] M. Toorani, A. Beheshti, Solutions to the GSM security weaknesses, in: 2nd International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST'08), IEEE, 2008, pp. 576–581, doi:10.1109/NGMAST.2008.88.
[54] C. Ioannou, V. Vassiliou, The impact of network layer attacks in wireless sensor networks, in: Secure Internet of Things (SIoT), 2016 International Workshop on, IEEE, 2016, pp. 20–28, doi:10.1109/SIoT.2016.009.
[55] I. Tomić, J.A. McCann, A survey of potential security issues in existing wireless sensor network protocols, IEEE Internet of Things Journal 4 (6) (2017) 1910–1923, doi:10.1109/JIOT.2017.2749883.
[56] F. Keblawi, D. Sullivan, Applying the common criteria in systems engineering, IEEE security & privacy 4 (2) (2006) 50–55.
[57] G. Baldini, A. Skarmeta, E. Fourneret, R. Neisse, B. Legeard, F.L. Gall, Security certification and labelling in internet of things, in: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), IEEE, 2016, pp. 627–632.
[58] G. Klein, J. Andronick, M. Fernandez, I. Kuz, T. Murray, G. Heiser, Formally verified software in the real world, Communications of the ACM 61 (10) (2018) 68–77.
[59] R. Anderson, S. Fuloria, Certification and evaluation: A security economics perspective, in: 2009 IEEE Conference on Emerging Technologies & Factory Automation, IEEE, 2009, pp. 1–7.
[60] M.B. Line, I.A. Tøndel, M.G. Jaatun, Current practices and challenges in industrial control organizations regarding information security incident management–does size matter? information security incident management in large and small industrial control organizations, International Journal of Critical Infrastructure Protection 12 (2016) 12–26, doi:10.1016/j.ijcip.2015.12.003.
[61] J. Liu, Y. Xiao, S. Li, W. Liang, C.P. Chen, Cyber security and privacy issues in smart grids, IEEE Communications Surveys & Tutorials 14 (4) (2012) 981–997, doi:10.1109/SURV.2011.122111.00145.
[62] F.M. Cleveland, Cyber security issues for advanced metering infrastructure (ami), in: Power and Energy Society General Meeting – Conversion and Delivery of Electrical Energy in the 21st Century, IEEE, 2008, pp. 1–5, doi:10.1109/PES.2008.4596535.
[63] V. Aravinthan, V. Namboodiri, S. Sunku, W. Jewell, Wireless ami application and security for controlled home area networks, in: Power and Energy Society General Meeting, IEEE, 2011, pp. 1–8, doi:10.1109/PES.2011.6038996.
[64] R.K. Bhatia, V. Bodade, Defining the framework for wireless-ami security in smart grid, in: International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), IEEE, 2014, pp. 1–5, doi:10.1109/ICGCCEE.2014.6921383.
[65] D. Grochocki, J.H. Huh, R. Berthier, R. Bobba, W.H. Sanders, A.A. Cárdenas, J.G. Jetcheva, Ami threats, intrusion detection requirements and deployment recommendations, in: 3rd International Conference on Smart Grid Communications (SmartGridComm), IEEE, 2012, pp. 395–400, doi:10.1109/SmartGridComm.2012.6486016.
[66] C. Brunschwiler, Wireless m-bus security, in: Black Hat USA, 2013, pp. 1–70. Whitepaper available at https://www.compass-security.com/fileadmin/Datein/Research/Praesentationen/blackhat_2013_wmbus_security_whitepaper.pdf
[67] D.R. Thomas, A.R. Beresford, A. Rice, Security metrics for the android ecosystem, in: 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices, 2015, pp. 87–98, doi:10.1145/2808117.2808118.
[68] A. Hahn, M. Govindarasu, Smart grid cybersecurity exposure analysis and evalution framework, Power and Energy Society General Meeting (2010) 1–6, doi:10.1109/PES.2010.5590152.
[69] A. Hahn, M. Govindarasu, Cyber attack exposure evaluation framework for the smart grid, IEEE Transactions on Smart Grid 2 (4) (2011) 835–843, doi:10.1109/TSG.2011.2163829.
[70] R. Fredriksen, M. Kristiansen, B.A. Gran, K. Stølen, T.A. Opperud, T. Dimitrakos, The CORAS framework for a model-based risk management process, in: S. Anderson, M. Felici, S. Bologna (Eds.), International Conference on Computer Safety, Reliability, and Security, Springer, 2002, pp. 94–105, doi:10.1007/3-540-45732-1_11.
[71] J. Jones, Factor analysis of information risk, 2004, US Patent App. 10/912,863.
[72] T. Kelly, R. Weaver, The goal structuring notation–a safety argument notation, in: Proceedings of the dependable systems and networks, 2004, p. 6.
[73] L. Cyra, J. Gorski, Support for argument structures review and assessment, Reliability Engineering & System Safety 96 (1) (2011) 26–37.
[74] S.E. Toulmin, The uses of argument, Cambridge university press, 1958.

Paper II

Criteria for Security Classification of Smart Home Energy Management Systems

Manish Shrestha, Christian Johansen, Josef Noll

Presented in First International Conference on Smart Information and Communication Technologies (SmartICT 2019) and Published by Springer in 2020 in the book entitled “Advances in Smart Technologies Applications and Case Studies” (Lecture Notes in Electrical Engineering Volume 684).

DOI: https://doi.org/10.1007/978-3-030-53187-4_19

A technical report (starting at page 79) supporting this publication was published in July 2019, with the research report number 492, ISBN 978-82-7368-457-8, and ISSN 0806-3036.

The report can be found at:

http://urn.nb.no/URN:NBN:no-79526

Criteria for Security Classification of Smart Home Energy Management Systems

Manish Shrestha[1,2], Christian Johansen[1], and Josef Noll[1]

[1] University of Oslo, Oslo, Norway, cristi@ifi.uio.no, josef@jnoll.net
[2] eSmart Systems AS, Halden, Norway, manishsh@student.matnat.uio.no

Abstract. Internet of Things (IoT) is a growing field and its use in home automation is no exception. However, the end users lack security awareness whereas the system designers lack the incentives for building secure IoT systems. To address this challenge, we propose the notion of security classes to assess and present the security of complex IoT systems both for the users and for developers. Furthermore, regulatory bodies can use our security classification method as a reference to derive requirements for adequate security. This paper extends the previous security classification methodology towards Smart Home Energy Management Systems (SHEMS). We demonstrate its applicability by performing a systematic security classification assessment of an industrial SHEMS. Results show that the use of security classes can give a good indication of security status and guidance to improve the security of IoT system.

Keywords: Security Classification, Exposure, Security assessment, Cybersecurity, Smart home, IoT

1 Introduction

The proliferation of IoT has created new transformative opportunities s.a. observed with smart homes [1, 14]. Today, the applications inside smart homes are more than luxury, where, e.g., energy management systems can enable efficient utilization of energy [4]. Industrial IoT providers are normally concerned with the development of functionalities, creating a range of communication and sensing capabilities integrated into small devices. However, security and privacy have been a major concern, which often hinders a wider adoption of IoT systems.

This paper is an extension of our previous work [12] where we introduced a general security classification methodology for smart grid systems. In this paper, we extend the security classes with details regarding connectivity classes and protection mechanisms suitable for Smart Home Energy Management Systems (SHEMS) and show the application of our approach to an existing commercial system. One motivation of the present work is to help companies to improve and maintain IoT security of their products guided by security classes and the protection mechanisms that they specify.

We describe in Section 2, the reference architecture for SHEMS that we follow, and briefly introduce the system from our case study. Our main contribution is presented in Section 3 where we extend the security classification method towards SHEMS. We show the application of this new methodology in Section 4 using a case study of existing SHEMS from Develco Products.

2 A Commercial Home Energy Management System

A SHEMS is a smart home system dedicated to saving energy by monitoring and managing electrical appliances, which may include load, storage, or generation resources [6, 7, 15]. Functional modules of SHEMS may include monitoring, logging, control, management, or alarm services [15]. Ghirardello et al. [5] summarize a smart home reference architecture (see Fig. 1) by integrating three different viewpoints: functional, physical, and communication. Based on this architecture, we describe the major components of smart home systems as below.

IoT Devices. These have as primary functions [5] to sense the environment, transfer data, and receive commands. As such, these have communication capabilities and may be able to interact with other components of the Home Area Network (HAN) such as IoT hubs, residential gateways, or other IoT devices. In SHEMS, IoT devices may include metering (and sensing) devices and controllable loads.

IoT Hub. It acts as a central controller of IoT devices as well as a bridge between these and the backend system. Sensor data is reported to the IoT hub, which translates and sends it to the backend system. Similarly, the IoT hub may receive control commands, which it can relay to the intended devices. Opposed to IoT devices, the IoT hub has considerably more computing capability and can make decisions to manage and control the IoT devices.

Residential Gateway. It is a bridge to connect IoT devices to the Internet [5], i.e., between the HAN and the Wide Area Networks (WAN). In some systems, a gateway may act as an IoT hub or vice versa.

Communication Channels. A SHEMS consists of two types of networks: HAN and WAN. The HAN is formed of the sensors and the IoT hub, and utilize wireless communication links s.a. Zigbee, Z-Wave, Wireless M-Bus, Thread [3, 5]. The IoT hub and devices may also utilize Wi-Fi or Ethernet to connect with the residential gateway. In a WAN, a SHEMS typically utilizes home internet provided by internet subscribers or cellular networks to communicate with the backend system.

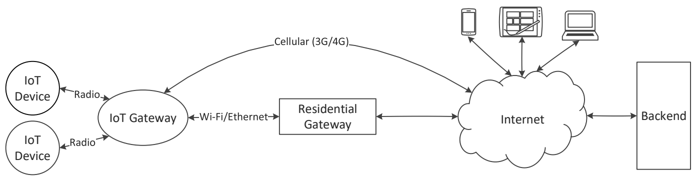


Fig. 1: Smart Home System Architecture.

Backend System. It is a centralized component, which manages several smart homes, and resides remotely, communicating with the IoT hub through the Internet and performing storage, monitoring, and control functionalities of IoT devices. Backend systems provide an interface to external applications through APIs, enabling communications with SHEMS [14].

Application and Network Data. The network data includes mainly information related to connectivity, whereas application data are those which actually have business value and include meter values, commands for controlling devices, log data, firmware image files, etc. Metered values are produced by IoT devices and sent to the IoT hub, which further sends these to the backend systems for storage and analysis. On the other hand, control commands are received by the IoT hub from the backend system and then sent to the IoT devices for execution.

We apply our security classification to the commercial smart home solution offered by E2U Systems AS, who use hardware provided by Develco Products and implement customized software solutions for smart homes. The Develco Products offer an IoT hub (called *Squid.link gateway*) and a variety of IoT devices such as smart plugs, sensors, alarms, meter interfaces, etc. The IoT hub is able to act as a residential gateway using the cellular network, and it also provides an Ethernet and a WLAN interface for Internet connection as well as a USB interface for plugging in 3G/4G dongles. The Squid.link gateway is a modular platform capable of bridging multiple wireless platforms, like Zigbee, Z-wave, Wireless M-Bus, in the HAN network. The wireless module on the main board of the IoT gateway communicates with the CPU using the SmartAMM protocol, which is the proprietary protocol that also facilitates communication between gateways and the backend system.

3 Extended Security Classification Method

The Smart Grid Security Classification (SGSC) methodology [12] is based on the ANSSI classification method [2]. However, instead of estimating the exposure based on the complexity of the system and attacker model (as in ANSSI), the SGSC combines the *connectivity* (which captures the surface of a system exposed to attacks) with *protection* (which describes the mechanisms of the system used to protect various part of the connectivity surface). Figure 2 summarises how a security class would then be computed. The computation first looks at the components of the system and then aggregates the results upwards until reaching the full system. Notably, the SGSC does not focus on attackers, as classical risk-based methods do, but is concerned instead with how a system can be securely built from the design phase. The benefit is that the SGSC helps system designers to choose the best security functionalities to meet their goal security class.

We consider two types of exposures: IT Exposure and Physical Exposure. For both, we evaluate the connectivity into one of five levels as follows:

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.

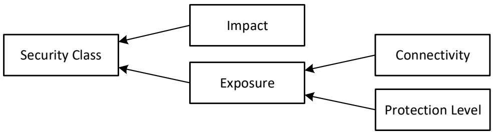


Fig. 2: Methodology of computing a security class [12] (Impact is as in ANSSI).

Table 1: Calculations of (a) Exposure Levels and (b) Security Classes

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

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.

We have defined Protection Levels (P) to capture the strength of security functionality implemented in a system. Protection Levels have been inspired by the Safety Integrity Levels (SIL) [10]. Instead of the attacker model, we consider the connectivity of the system when setting the required security mechanisms. Each security mechanism possesses a different strength level which can be ranked. We have defined five protection levels, where P1 represents no protection and P5 represents the strongest protection mechanisms. Table 1(a) shows the evaluation of exposure level from connectivity and protection level. The evaluation of the protection level is conducted by security experts.

In [12] we have not considered protection mechanisms in detail (the same as how standards like ANSSI also do). In this paper, we detail this important part

Table 2: Referred sources for the construction of security criteria.

Protection Criteria	**Source**
Data Encryption	ISO 27002, OWASP, ETSI
Communication and Connectivity Protection	IIC, ISO 27002, ETSI
Software/Firmware Security	ISO 27002, OWASP, ETSI
Hardware-based Security Controls	CSA
Access Control	ISO 27002, OWASP, IIC, CSA, ETSI
Cryptography Techniques	IIC, ISO 27002
Physical and Environmental Security	ISO 27002, OWASP, CSA
Monitoring and Analysis	ISO 27002, OWASP, IIC, CSA, ETSI

of our SGSC by ranking various security functionalities, focusing on our SHEMS application domain. Table 2 lists classes of functionalities.

We extend [12] by extracting the security criteria for evaluating protection levels based on the following standards and best practices:

ISO 27002 which however does not cover the IoT systems;
CSA the IoT Working Group of the Cloud Security Alliance;
IIC the Industrial Internet of Things Volume G4 Security Framework;
OWASP "IoT Security Guidance"; and
ETSI TS 103 645 "Cyber Security for Consumer Internet of Things".

We detail further the security criteria with security functionalities inspired by the IoT Security Compliance Framework proposed by IoT Security Foundation (IoTSF), which is in the form of a checklist. Table 3 shows the mapping of security criteria to security functionalities and protection level.

4 Applying the Security Classification to SHEMS

The SHEMS in our case complies with the reference architecture from Section 2 and consists of a centralized IoT hub and smart plugs connected to controllable loads s.a. water heater, air conditioner, floor heating, etc. For simplicity, we do not include the storage batteries that can act as both load and generation device. We first identify the criticality (**Impacts**) of successful cyberattacks on SHEMS.

Safety. Leakage of data from SHEMS may disclose the presence of people inside their house, which may result in burglary or worst. Moreover, residents may feel unsafe (reducing trust in SHEMS) if they realize they are being watched.

Grid imbalance. During the execution of a demand response program, devices that utilize higher energy are turned off to shave the peaks. If an attacker can switch on/off a large number of loads, these may use unexpected amounts of energy that may destabilize the grid [8, 13].

Increased electricity bills. Compromising SHEMS may result in equipment being switched on without authorized persons noticing.

Privacy. Data from SHEMS can be privacy sensitive, e.g., [9] have demonstrated that mere high-frequency consumption data can be exploited to derive private information s.a. number of people in the house, sleep routines, the presence of babies at home, etc. Compromised SHEMS data may contain even more detailed information. Stealing such data may result in the exposure of personal habits of the residents, which may impact social reputation.

Agents for other cyberattacks. Typically, smart home gateways have connectivity to the Internet. A compromised gateway may act as a bot to launch several other attacks.

Among the aforementioned impacts, grid imbalance and agents for other cyberattacks can be considered as major impacts as these may result in blackouts and damage of physical infrastructures. The remaining impacts could be considered moderate or minor.

We limit the presentation of the application of security classification only to Application and Network Data, in particular, we assess the Command and

Table 3: Protection Level Requirements

Protection Criteria	**Security Functionality**	**P5**	**P4**	**P3**	**P2**
Data Encryption	Encryption of data between system components	x	x	x	x
	Strong encryption mechanism	x	x	x	
	Credentials should not be exposed in the network	x	x	x	
	End-to-end encryption	x	x		
	Should not use custom encryption algorithms	x	x		
	Sensitive stored data should be encrypted	x	x		
Communication and Connectivity Protection	Have a minimal number of network ports open	x	x	x	
	Devices should not be accessible from the Internet	x	x	x	
	Only authorized components can join the network	x	x	x	
	Use only standard communication protocol	x	x		
Software /Firmware Security	Updatability of device firmware	x	x		
	Updatability of the operating system	x	x		
	Automatic updates available	x	x		
	Encryption of update files	x	x		
	Signing update files before installing	x	x		
Hardware-based Security Controls	Using Trusted Platform Modules (TPM)	x	x		
	Use of Memory Protection Units (MPUs)	x	x		
	Incorporate Physically Unclonable Functions	x	x		
	Use of Cryptographic Modules	x	x		
Access Control	Disable remote access functionality	x			
	Only authorized devices can join the network	x	x	x	
	Default and weak passwords should not be used	x	x	x	
Cryptography Techniques	Secure bootstrapping	x	x		
	Secure key generation	x	x		
	Secure key storage	x	x		
	Secure key distribution	x	x	x	
	Secure key rotation	x	x		
	Message integrity	x	x	x	
Physical and Environmental Protection	Tamper resistance	x	x		
	Minimal physical ports available	x	x	x	
	Physical security of connections	x	x	x	
	Ability to disable external ports and only minimal ports enabled	x	x		
	Only authorized physical access	x	x	x	
Monitoring and Analysis	Monitoring system components	x	x		
	Analysis of monitored data	x	x		
	Act on analysed data	x			

Control (C&C) for a demand response program, which is one of the most critical components of SHEMS. We apply the classification method in two scenarios.

Scenario I: Centralized Control. In this scenario, Distribution System Operators (DSO) have an agreement with consumers to control the SHEMS appliances to properly manage peaks of energy demand. In our system, each controllable device is plugged into the corresponding smart plug and depending on the device and their maximum effect, rules for controlling them are defined, e.g., a water heater with a maximum capacity 3kW can be controlled only between 8:00 AM to 6:00 PM during weekdays, and once turned off, it cannot be turned on for minimum 15 minutes. The DSOs forecast the energy demand in advance and if reductions are needed at given times, DSOs optimally select the devices to be turned off for a given duration to meet the goal of targeted reduction of consumption and control commands are sent to the selected devices.

Class Evaluation. The connectivity between the IoT device and the hub is C3 (cf. Section 3) and between the hub and the backend system is C5. If an attacker is able to manipulate the device control only inside the HAN (C3), the impact is only Minor. However, if an attacker is able to trigger or manipulate the message for the demand control program from the backend (C5), several devices can be turned off, resulting in grid imbalance as discussed above. As a result, for this scenario, we evaluate the overall impact as Major.

To evaluate the security class, we first select the relevant security criteria for C&C as Data Encryption, Communication and Connectivity Protection, Access Control, and Monitoring and Analysis. We then evaluate the protection level based on the strength of the security functionalities in the selected criteria. Due to space limitations, we do not discuss here our specific evaluations, but provide details in the technical report [11]. Using Table 3 we assign the overall protection level P4.

Using Table 1(a) we determine from the computed values of connectivity (C5) and protection level (P4), the exposure E3. Using Table 1(b) we get the class D (Impact Major and Exposure E3), which is a poor score not suitable for SHEMS. To improve the security class, Table 1(b) indicates that either exposure or impacts need to be reduced. Similarly, exposure can be reduced either by increasing the protection level or by reducing the connectivity, cf. Table 1(a).

Scenario II: Edge Control. In this scenario, the control signals are sent by the IoT hub autonomously, based on the time of peak demand or price of electricity, and thresholds set by the end-user. Users can also set priorities for the devices that need to be controlled and rules to decide e.g., when and how long the devices can be controlled. Thresholds and rules can also be persisted in the IoT gateway so to control the devices without requiring interaction with the backend.

Class Evaluation. Similarly, if an attacker can manipulate the control message within the HAN network (C3), the impact is considered as Minor. However, since there is no flow of commands from the backend system, an attacker cannot influence many devices on a large scale. Since there are no changes in the protection mechanisms, we can consider it as P4. Moreover, using Table 1(a), we obtain the Exposure E2 and using Table 1(b) we computed the security class as A.

The analyses of scenario I and II showed that by moving from the centralized control to the edge control for the demand control functionality, the security class of the demand control is significantly improved from class D to class A. In addition, scenario II may even be more efficient and have lower latency because the trigger of device control initiates locally rather than from the backend system to several IoT devices. Such improvements in the design of IoT systems should be considered to improve the security of the overall system.

5 Conclusion and Further Work

We present the security classification methodology extended with details regarding security functionalities relevant for SHEMS. We have applied this methodology to the commercial SHEMS from our collaborators E2U, and presented in this paper how we performed the assessment of the component for control and command of the SHEMS within demand and response programs. We have first evaluated the security class of the C&C for a centralized control architecture and then saw that the classification methodology can give indications of the possible changes in the design of the system to improve the security class (as in our second scenario). Further work can focus on aggregation mechanisms for calculating the overall system security class from its components.

References

1. Aldrich, F.K.: Smart homes: past, present and future. In: Inside the smart home, pp. 17–39. Springer (2003)
2. ANSSI: Classification Method and Key Measures (2014)
3. Celebucki, D., Lin, M.A., Graham, S.: A security evaluation of popular internet of things protocols for manufacturers. In: ICCE, pp. 1–6. IEEE (2018)
4. Fitriaty, P., Shen, Z., Sugihara, K.: How green is your smart house: Looking back to the original concept of the smart house. In: Green City Planning and Practices in Asian Cities, pp. 39–76. Springer (2018)
5. Ghirardello, K., Maple, C., Ng, D., Kearney, P.: Cyber security of smart homes: Development of a reference architecture for attack surface analysis (2018)
6. Lee, J.I., Choi, C.S., Park, W.K., Han, J.S., Lee, I.W.: A study on the use cases of the smart grid home energy management. In: ICTC, pp. 746–750. IEEE (2011)
7. Liu, Y., Qiu, B., Fan, X., Zhu, H., Han, B.: Review of smart home energy management systems. Energy Procedia **104**, 504–508 (2016)
8. Mohsenian-Rad, A.H., Leon-Garcia, A.: Distributed internet-based load altering attacks against smart power grids. IEEE Trans. Smart Grid **2**(4), 667–674 (2011)
9. Molina-Markham, A., Shenoy, P., Fu, K., Cecchet, E., Irwin, D.: Private memoirs of a smart meter. In: Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building, pp. 61–66. ACM (2010)
10. Redmill, F.: Understanding the use, misuse and abuse of safety integrity levels. In: 8^{th} Safety-critical Systems Symposium, pp. 8–10 (2000)
11. Shrestha, M., Johansen, C.: Criteria for Security Classification of Smart Home Energy Management Systems (long version). Tech. Rep. 476, Uni. Oslo (2019)
12. Shrestha, M., Johansen, C., Noll, J., Roverso, D.: A Methodology for Security Classification applied to Smart Grid Infrastructures. International Journal of Critical Infrastructure Protection (IJCIP) (2019). (to appear)
13. Soltan, S., Mittal, P., Poor, H.V.: Blackiot: Iot botnet of high wattage devices can disrupt the power grid. In: 27th USENIX Security Symposium, pp. 15–32 (2018)
14. Stojkoska, B.L.R., Trivodaliev, K.V.: A review of internet of things for smart home: Challenges and solutions. Journal of Cleaner Production **140**, 1454–1464 (2017)
15. Zhou, B., Li, W., Chan, K.W., Cao, Y., Kuang, Y., Liu, X., Wang, X.: Smart home energy management systems: Concept, configurations, and scheduling strategies. Renewable and Sustainable Energy Reviews **61**, 30–40 (2016)

UiO : Department of Informatics
University of Oslo

Criteria for Security Classification of Smart Home Energy Management Systems (long version)

Manish Shrestha , Christian Johansen , Josef Noll

Research report 492, July 2019

ISBN 978-82-7368-457-8
ISSN 0806-3036

Criteria for Security Classification of Smart Home Energy Management Systems (long version)

Manish Shrestha   Christian Johansen   Josef Noll

July 2019

Abstract

Internet of Things (IoT) is a growing field and its use in home automation is one of the dominating application areas. The heterogeneity and limited capacity of storage and processing power make the security of IoT systems challenging. Besides, the end users lack security awareness and the system designers lack the incentives for building secure IoT systems. To address this challenge, we propose the notion of security classes to assess and present the security of complex IoT systems both for the end users and for developers. Furthermore, regulatory bodies can use our security classification method as a reference to derive requirements for adequate security. This report presents a security classification methodology and extends it for the Smart Home Energy Management Systems (SHEMS). We demonstrate its applicability by performing a systematic security classification assessment of an industrial SHEMS. Results show that the use of security classes is a good indication of the level of security, as well as a guide to improve the security of IoT systems.

This technical report is a long version of the conference paper [28].

[0] *Address for correspondence:*
Department of Informatics, University of Oslo, P.O. Box 1080 Blindern, 0316 Oslo, Norway.
E-mail: `cristi@ifi.uio.no`

Contents

1 Introduction

The proliferation of the Internet of Things (IoT) has created new transformative opportunities. One good example can be observed within smart homes [2, 31]. Aldrich defines the concept of *smart homes* as “a residence equipped with computing and information technology which anticipates and responds to the needs of the occupants, working to promote their comfort, convenience, security and entertainment through the management of technology within the home and connections to the world beyond” [2]. Today, the applications inside smart homes are more than luxury, where, e.g., energy management systems can enable efficient utilization of energy [14]. Modern smart home systems are composed of several sensors and actuators that communicate with a central hub called gateway, which might connect the sensor network to a cloud system.

Industrial IoT providers are usually driven by the development of functionalities, creating a range of communication and sensing capabilities integrated into small devices. However, from the customer’s point of view, security and privacy has been a major concern, and often hinders a wider adoption of IoT systems. According to Symantec’s Internet Security Threat Report (ISTR) 2018, the number of cyberattacks on IoT devices has increased by 600% between the years 2016 and 2017.[1] This is an indication that current security practices need to evolve to fit IoT systems, which is formed by the integration between several heterogeneous components.

This report is an exemplification and enhancement of our previous work [29] where we have introduced a general security classification methodology for smart grid systems and Advanced Metering Infrastructures (AMI). In this report, we extend the security classes with details regarding connectivity classes and protection mechanisms suitable for Smart Home Energy Management Systems (SHEMS) and show the application of our approach to an existing commercial system from one of our partner companies E2U Systems AS. One motivation of the present work is to help companies to improve and maintain IoT security of their products guided by security classes and the protection mechanisms that they specify.

We describe in Section 2, the reference architecture for SHEMS that we follow, and briefly introduce the system from our case study. We also describe the major communication standards used in SHEMS. Our main contribution is presented in Section 3 where we extend the security classification method towards SHEMS. We show the application of this new methodology in Section 4 using a case study of existing SHEMS from Develco Products. We consider two alternatives for device control mechanism for demand response activities and evaluate their class to illustrate the applicability of security class methodology. Section 5 concludes our work with an overview of future work.

2 A Commercial Home Energy Management System

This section provides a brief overview of typical smart home systems describing its components. Next, we take a real example of one of the smart home solution provided by Develco Products and describe its architecture and how such systems are being used to provide practical solutions in the industry. The main goal is to develop a certification scheme which can be used also internally within an IoT company to specify baseline security requirements and maintain the security level over time.

[1]Symantec Corporation, “Internet Security Threat Report(ISTR), Volume 23”. https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf.

2.1 SHEMS Reference Architecture

A SHEMS is a smart home system dedicated to saving energy by monitoring and managing electrical appliances, which may include load, storage, or generation resources [19,20,32]. Heat pumps are typical examples of load resources. Similarly, car batteries and solar panels are examples of storage and generation resources respectively. Functional modules of SHEMS may include monitoring, logging, control, management, or alarm services [32]. Ghirardello et al. [16] summarize a smart home reference architecture (see Fig. 1) by integrating three different viewpoints: (i) functional viewpoint, (ii) physical viewpoint, and (iii) communication viewpoint. The functional viewpoint focuses on the functionality of the IoT network in a smart home environment. Similarly, the physical viewpoint is concerned with the physical components involved to meet the functionality. The communication viewpoint focuses on the communication technology that enables interactions between the physical components. Thus, this framework gives an overview of the physical components and the interactions between them.

Based on this architecture, we describe the major components of smart home systems below.

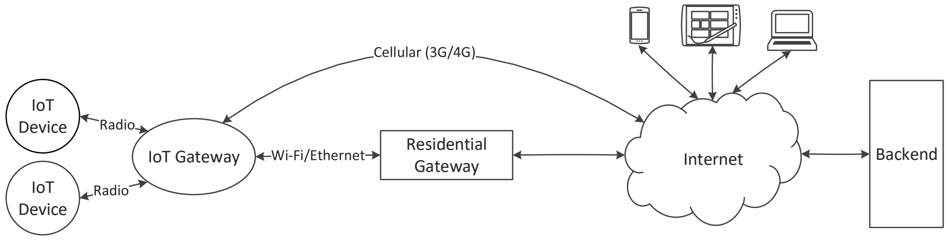


Figure 1: Smart Home System Architecture.

A SHEMS consists of IoT devices, IoT Hubs, residential gateways, clients (including smartphones, tablets, computers and software applications to utilize the services to the consumers), communication channels, backend systems, and Application and Network Data (AND). These components utilize wired or wireless communication channels to communicate with the cloud, as depicted in Figure 1.

IoT Devices. These have as primary functions [16] to sense the environment, transfer data, and receive commands. As such, these have communication capabilities and may be able to interact with other components of the Home Area Network (HAN) such as IoT hubs, residential gateways, or other IoT devices. In SHEMS, IoT devices may include metering (and sensing) devices and controllable loads. Some examples of smart home devices include humidity sensors, heat alarms, motion sensors, meter interfaces, smoke detectors, window-sensors, thermostats, smart plugs and light dimmers.

IoT Hub. It acts as a central controller of IoT devices as well as a bridge between these and the backend system. Sensor data are reported to the IoT hub, 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. Opposed to IoT devices, the IoT hub has considerably more computing capability and can make decisions to manage and control the IoT devices.

Residential Gateway. The smart home gateway connects sensors and actuators to the backend system through the Internet. In other words, it is a bridge between the HAN and the Wide Area Networks (WAN). Quite often an IoT hub and residential gateway functionalities are integrated into one device.

Communication Channels. A SHEMS consists of two types of networks: HAN and WAN. The HAN is formed of the sensors and the IoT hub, and utilize wireless communication links such as Zigbee, Z-Wave, Wireless M-Bus, Thread [9, 16]. The IoT hub and devices may also utilize Wi-Fi or Ethernet to connect with the residential gateway. In a WAN, a SHEMS typically utilizes home internet provided by internet subscribers or cellular networks to communicate with the backend system.

Backend System. It is a centralized component, which manages several smart homes, and resides remotely, communicating with the IoT hub through the Internet and performing storage, monitoring, and control functionalities of IoT devices. Backend systems provide an interface to external applications through APIs, enabling communications with SHEMS [31].

Application and Network Data. The network data includes mainly information related to connectivity, whereas application data are those which actually have business value and include meter values, commands for controlling devices, log data, firmware image files, etc. Metered values are produced by IoT devices and sent to the IoT hub, which further sends these to the backend systems for storage and analysis. On the other hand, control commands are received by the IoT hub from the backend system and then sent to the IoT devices for execution.

2.2 Major Communication Standards

Here we describe three major communication standards used in Commercial SHEMS for HAN.

2.2.1 Zigbee

Zigbee is a low-cost, lower-power consuming, two-way wireless communication standard from Zigbee Alliance. It enhances the IEEE 802.15.4 standard by using the Network layer and Application layer to define additional communication features. It uses the open trust model where each network layer in the protocol stack trusts each other. There are three types of nodes in a Zigbee network: coordinator, router, and end-device. A Zigbee network has only one coordinator and acts as a parent of all the nodes in the network. A coordinator allows other nodes to join the network by selecting an appropriate channel, frequency, and PAN id of the network. A router node is used to route traffic between different nodes. However, an end device does not route any traffic, as it simply sends or receives messages from a router or a coordinator.

Zigbee devices provide access control, data encryption, data integrity, and replay protection [11]. A Zigbee network supports three types of encryption keys: master key, network key, and the link key. A master key is generally used for exchanging link keys. Master keys are usually pre-installed, whereas some systems may have a key-load mechanism where a dedicate key-load key (derived from link key) is used to protect the master key during transport from the trust center to the device in the network. Zigbee uses AES (Advanced Encryption Standard) 128-bit encryption for exchanging messages.

To simplify the interoperability of devices, Zigbee provides, by design, the same security level for all devices in all the layers in the network [3]. Therefore, if there is one device in the network that does not support a higher level of security, then the security of the whole network is downgraded to a lower level of security. To overcome this issue, either critical devices should have a separate network or all devices in the network should support the required level of security. Table 1 shows security levels available for network and application layers in Zigbee.

Table 1: Security Levels in Zigbee Network and Application Layers [3]

Security Level Identifier	**Security Attributes**	**Data Encryption**	**Frame Integrity (length M of Message Integrity Code (MIC), in Number of Octets)**
0x00	None	OFF	NO (M=0)
0x01	MIC-32	OFF	YES(M=4)
0x02	MIC-64	OFF	YES(M=8)
0x03	MIC-128	OFF	YES(M=16)
0x04	ENC	ON	NO
0x05	ENC-MIC-32	ON	YES(M=4)
0x06	ENC-MIC-64	ON	YES(M=8)
0x07	ENC-MIC-128	ON	YES(M=16)

Though Zigbee offers strong encryption mechanisms, failing to protect the keys may result in security breaches [11]. Replay protection is based on the frame counter; if the latest counter is less than the last received counter, the message is ignored. An attacker could modify the message by increasing the counter [13] and then launch replay attacks. Proper key management, tamper resistance/detection and replay protection mechanisms can elevate security in Zigbee networks [12, 13, 33].

2.2.2 Z-Wave

Z-Wave is based on the G.9959 specification from ITU which specifies the Physical and MAC layer. Like Zigbee, it consumes little power and has long battery life. A Z-Wave network includes two types of nodes based on their roles: controller and slave. The controller sends commands to the slaves and is responsible to add or remove slaves from the network, having a full overview of the network. Slave nodes are the sensors and actuators that reply and execute the commands from the controller node. These are also capable of forwarding the commands to other nodes.

An open source implementation of Z-Wave called OpenZWave is available. It is based on public information and reverse engineering.[2] Currently, Z-Wave specifications are made available as a public standard at http://zwavepublic.com. Since most of the details of Z-Wave was not open until 2016, very few security assessments are publicly available.

Z-Wave uses AES 128-bit encryption for data security. It is backward compatible and is highly interoperable. The security framework “S2” of Z-Wave supports Diffie-Hellman key exchange, which makes the key exchange process more secure than in Zigbee. However, Z-Wave supports older devices that do not support encrypted and authenticated communications. This weakens the security of the Z-Wave network [1]. Moreover, one can also downgrade the security process to the previous version and exploit its vulnerabilities to compromise the network.[3]

The work of [15] found implementation issues of Z-Wave, where all 16 bytes of the temporary key used to exchange the encryption key were all zeros, which allowed to decrypt the encryption key. It was also observed that there was no state validation in the key exchange protocol. Therefore, the slave does not know if the key derivation has already been performed. Thus,

[2]“OpenZWave”. https://github.com/OpenZWave/open-zwave.

[3]“Z-Shave. Exploiting Z-Wave downgrade attacks”. https://www.pentestpartners.com/security-blog/z-shave-exploiting-z-wave-downgrade-attacks/. . Accessed: October 26, 2018

one could spoof the controller and request a new key derivation process, which can reset the established network key on the target device and issue unauthorized commands to the slave devices.

2.2.3 Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) is a low-power, low-cost wireless protocol that supports frequency-hopping over 40 channels.[4] BLE also supports multiple network topologies. Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) is another version of Bluetooth that supports only point-to-point network topologies and is optimized for continuous data streaming. Since BLE is designed to be scalable for larger networks, it is suitable for home automation systems. BLE typically supports four association models:

- *Numeric Comparison*: In this model, the user is shown a 6-digit number on both displays and then asked to confirm whether the numbers are identical, thus protecting the network from MITM (man-in-the-middle) attacks.
- *Just Works*: This model also uses Numeric Comparison, but the user is never shown a number, and the application may accept the connection. This model is prone to MITM attacks.
- *Out of Band*: This model uses Out of Band mechanisms to discover the devices and exchange the keys for the pairing process.
- *Passkey Entry*: This model is used when one of the devices has the input capability but does not have a display. Then, the user will be displayed the 6-digit passkey in one device and ask another device to enter it.

BLE provides five basic security services, namely Authentication, Confidentiality, Authorization, Message Integrity and Pairing/Bonding [26]. BLE also preserves privacy by changing the device addresses frequently, making it difficult to track the devices on the network [9]. BLE supports two security modes, Mode 1 and Mode 2. Mode 1 has four levels of security:

1. No security (No authentication and no encryption)
2. Unauthenticated pairing with encryption
3. Authenticated pairing with encryption
4. Authenticated LE Secure Connections pairing with encryption using a 128-bit strength encryption key.

The higher level of security satisfies the lower levels. For example, level 2 security satisfies level 1, level 3 satisfies level 2, and so on.

Similarly, LE Security Mode 2 has two security levels:

1. Unauthenticated pairing with data signing
2. Authenticated pairing with data signing

Vendors could also select which security mode their devices will operate in and with what level within that mode. If vendors do not choose level three they will lose either data encryption, integrity, authentication or a combination of three.

[4]"Radio Versions". https://www.bluetooth.com/bluetooth-technology/radio-versions.

Security of BLE highly depends on the configuration selected for the system. Bluetooth 4.2 and onwards has support for Elliptic Curve Cryptography (ECC). However, ECC in legacy mode pairing is vulnerable to MITM [9]. As BLE was designed for star topology networks, data channels are protected only for a single hop. Hence, BLE does not support end-to-end encryption and authentication in mesh networks [10, 26]. Thus, additional security controls should be provided on top of the Bluetooth stack [26]. The work of [18] pointed out that the Temporary Key (TK) which is used to generate the encryption key can be brute-forced in less than 20 seconds because of its short length. Thus, they proposed to increase the length of TK to improve BLE security.

2.3 Comparisons

In Table 2 we compare the above three standards, which are meant to be used in home automation domain. In this table, for simplicity, we have not considered the configuration of the system for computing security classes and assumed that they have the best configuration. Otherwise, considering different configurations may result in a different table for different settings of security attributes. For instance, if the security level for Zigbee is set to level 4, then it will only have encryption but no frame protection mechanism, which will eventually degrade the protection level and will eventually affect the security class.

In our case, Z-Wave and BLE have a higher level of protection than Zigbee in terms of key exchange as they use asymmetric encryption, as opposed to symmetric encryption used in Zigbee. In the case of BLE, end-to-end encryption is not supported. However, it can be protected by using a star topology instead of a mesh network topology. Moreover, Z-Wave specifications are not completely open, which should also be taken into consideration when selecting the right technology. In our case, over the air upgrade is supported only for Zigbee and thus, though Zigbee does not support asymmetric encryption for key exchange, we select the Zigbee supported system for our case study.

Table 2: Comparison of major security functionalities in Zigbee, Z-Wave, and BLE.

Security Mechanism	**Protection Level**		
	Zigbee	**Z-Wave**	**BLE**
Data Encryption	Y	Y	Y
End-to-end Encryption	Y	Y	N
Node authentication	Y	Y	Y
Key exchange	Y (symmetric)	Y (asymmetric)	Y (asymmetric)
Integrity Protection	Y	Y	Y

2.4 System Description

As a case study, we apply our security classification to the commercial smart home solution offered by E2U Systems AS using hardware provided by Develco Products and implement customized software solutions for smart homes.[5] Develco Products[6] focuses on smart home and smart energy domain and delivers a wireless infrastructure platform for solution providers. In this section, we briefly describe the smart home solution provided by E2U Systems.

The Develco Products offer a Linux based IoT hub called *Squid.link gateway* (Fig. 2) consisting of ARM9TDMI, 454 MHz CPU (Central Processing Unit), and a variety of IoT

[5]"E2U Systems". https://e2usystems.com.

[6]https://www.develcoproducts.com

devices such as smart plugs, sensors, alarms, meter interfaces, etc. The IoT hub is able to act as a residential gateway using the cellular network, but it also provides an Ethernet and a WLAN interface for Internet connection as well as a USB interface for plugging in 3G/4G dongles. The Squid.link gateway is a modular platform capable of bridging in the HAN network multiple wireless platforms, like Zigbee, Z-wave, Wireless M-Bus, Bluetooth Low Energy. We have considered Zigbee for HAN network during our analysis. The wireless module on the main board of the IoT gateway communicates with the CPU using the SmartAMM protocol, which is a proprietary protocol that also facilitates communications between gateways and the backend systems.

Before installing the smart home system into the households, it needs to be pre-configured, which includes registration of gateways and its devices in the backend system, so that a device can only communicate with the gateway to which it is registered. Unregistered or unknown devices cannot join the HAN network. Pre-configuration also involves the configuration of the gateway to communicate with the proper backend system.

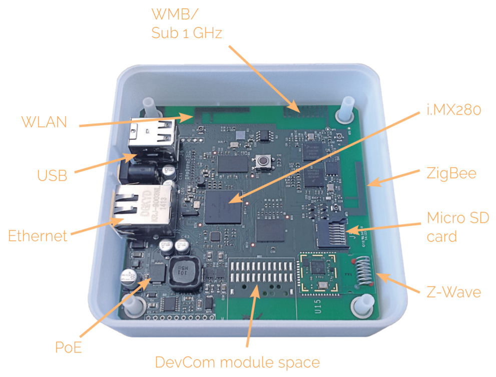


Figure 2: Squid.link Gateway

3 Extended Security Classification Method

The French "Agence Nationale de la Ssécurité des Systémes d'Information" (ANSSI) proposed a classification method for standardizing security measures for Industrial Control System (ICS) [4]. This classification is based on established risk analysis methods and provides general guidelines to determine the security class of an ICS. However, the computation of exposure in the ANSSI method does not fit smart grid systems, nor smart homes for the same matter. Therefore, we have proposed the Smart Grid Security Classification (SGSC) method [29], which extends the ANSSI classification method. However, instead of estimating the exposure based on the complexity of the system and attacker model (as in ANSSI), the SGSC combines the *connectivity* (which captures the surface of a system exposed to attacks) with *protection* (which describes the mechanisms of the system used to protect the connectivity surface). SGSC considers two major factors: Impact and Exposure. Impact indicates how critical a given subsystem is, whereas Exposure shows what functionality surface does it provide to be attacked.

Figure 3 summarises how a security class is computed. The computation first looks at the components of the system and then aggregates the results upwards until reaching the security classification of the whole. Notably, the SGSC does not focus on attackers, as classical risk-based methods do, but is concerned instead with how a system can be securely built from the design point of view. The benefit is that the SGSC helps system designers to choose the most appropriate security functionalities to meet the envisaged security class. In addition, the focus on "secure-by-built" systems is better suited for long term applicability, as threats and vulnerabilities only represent a snapshot, whereas security classes present an inherent view of an IoT system.

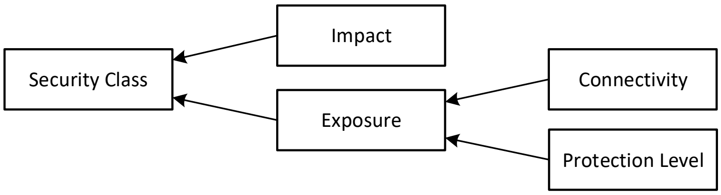


Figure 3: Methodology of computing a security class [29] (Impact as used by ANSSI).

In SGSC, we use five levels of impacts namely *Insignificant, Minor, Moderate, Major* and *Catastrophic*. Classifying impact is specific for the system under evaluation and is open to the judgement of security experts. We define exposure as the degree to which system's interfaces are available to attacks. There are two types of exposures: IT Exposure and Physical Exposure. Exposure is the result of Connectivity and Protection Level.

For both, we evaluate the connectivity into one of five levels as follows:

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.

We have defined Protection Levels (P) to capture the strength of security functionality implemented in a system. Protection Levels have been inspired by the Safety Integrity Levels (SIL) [27]. SIL is the number assigned to the safety function of a given system.[7,8] SIL broke the belief that the safety of a system is binary and is either safe or not. It introduced *levels* for safety [27]. Using a similar approach, Security Levels were introduced in the ISA-99 standard, which later changed to Security Assurance Levels (SALs). SAL is influenced by SIL but the levels of SAL are based on the strength of attackers. Because security systems have much broader application, consequences and possible circumstances, [17] claim that representing security assurance level by a mere number is not enough, and therefore, propose a vector approach to describe security requirements.

We use a similar approach as SAL by introducing Protection Category (PC). However, instead of the attacker model, we consider the connectivity of the system when setting the required security mechanisms. We have defined five levels of protection categories to represent the increasing scope of security functionality, which is as follows [29]:

[7]"SIL Made Simple". http://www.valve-world.net/pdf/vw10ce_actuation_-cameron.pdf

[8]"Practical Overview of Implementing IEC 62443 Security Levels in Industrial Control Applications". Schneider Electric white paper.

PC 1: Includes Physical and Environmental Protection

PC 2: Includes PC 1 and Network Protection

PC 3: Includes PC 2 and Wireless Protection

PC 4: Includes PC 3 and Private Infrastructures protection

PC 5: Includes PC 4 and Cloud protection

PC represents the goal of protecting a given type of setup by using security functionalities (or mechanisms). Each security mechanism possesses a different strength level, which can be ranked. We have defined five protection levels, where P1 represents no protection and P5 represents the strongest protection mechanisms. Below are the guidelines to determine the protection levels:

Protection Level 1 (P1) : This level includes systems that have no security mechanisms.

Protection Level 2 (P2) : This level possesses basic security features and has little impact on improving security. Security functionalities implemented are easy to break, e.g., WEP (Wired Equivalent Privacy) password.

Protection Level 3 (P3) : This level of protection provides advanced security concepts, e.g., WPA (Wireless Protected Access) passwords.

Protection Level 4 (P4) : Protection level 4 possesses advanced security concepts, including also basic monitoring capabilities.

Protection Level 5 (P5) : This level possesses state-of-the-art protection mechanisms and advanced monitoring capabilities. Additionally, there are no vulnerabilities and issues discovered in any of the component or security mechanism. If a new security issue is discovered in a system/subsystem/component, it can no longer have protection P5. It will rather degrade to lower levels.

Table 3 shows the evaluation of exposure level from connectivity and protection level. The evaluation of the protection level is conducted by security experts.

Table 3: Calculation 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

A security class can be expressed in terms of impact and exposure levels. A security class represents the quality of security of a given system and is represented by letters A to F, where A represents the highest security class and F being the lowest. A higher security class means either the impact is low, or the exposure is low. If the impact is high, the exposure must be reduced to obtain a higher class. Thus, a security class can be improved by lowering either exposure or impact, or both (see table 4). Security classes can be used not only for determining the security status of the system, but also to define the appropriate security requirements for a given system. Thus, it can be used to make purchase decisions on smart home systems for residents and also for regulatory bodies to have control over the security standard of IoT systems.

Table 4: Calculation 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

In our earlier work, we have not considered protection mechanisms in detail (the same as how standards like ANSSI also do) [29]. In this report, we detail this important part of our SGSC by ranking various security functionalities, focusing on our smart home application domain. Table 5 lists classes of functionalities and their respective sources.

Table 5: Referred sources for the construction of security criteria.

Protection Criteria	**Source**
Data Encryption	ISO 27002, OWASP, ETSI
Communication and Connectivity Protection	IIC, ISO 27002, ETSI
Software/Firmware Security	ISO 27002, OWASP, ETSI
Hardware-based Security Controls	CSA
Access Control	ISO 27002, OWASP, IIC, CSA, ETSI
Cryptography Techniques	IIC, ISO 27002
Physical and Environmental Security	ISO 27002, OWASP, CSA
Monitoring and Analysis	ISO 27002, OWASP, IIC, CSA, ETSI

The goal of this research is to help industries establish and maintain good security standards of their systems by providing guidelines and strategies to improve overall system security. Understanding the sources of threat and the impacts is one way to understand the criticality of the system [6]. For instance, control command on an autonomous vehicle is much more critical than the control command to switch off the lights of a bedroom. We extend [29] by extracting the security criteria for evaluating protection levels based on the following standards and best practices:

ISO 27002 This standard provides general guidelines on information security management. Like other standards, it is also highly document-oriented. However, it does not cover the IoT systems until now [9].

Cloud Security Alliance (CSA) The IoT Working Group of CSA provides 13 step guidelines and considerations to secure IoT products targeted towards the developers of IoT devices [10].

[9]ISO/IEC 27002:2013 Information technology – Security techniques – Code of practice for information security controls (second edition).

[10]IoT Working Group, Cloud Security Alliance (CSA), "Future-proofing the Connected World: 13 Steps to Developing Secure IoT Products". https://downloads.cloudsecurityalliance.org/assets/research/internet-of-things/future-proofing-the-connected-world.pdf

Industrial Internet of Things (IIC) The Industrial Internet of Things Volume G4 Security Framework provides the Business and Functional viewpoints towards IoT security [11]. It describes the architecture of IoT systems and provides guidance to create secure IoT systems.

OWASP "IoT Security Guidance" provides general guidelines to the manufacturers, developers, and consumers to secure their IoT products [12].

ETSI TS 103 645 is a technical specification entitled "Cyber Security for Consumer Internet of Things" that provides cybersecurity guidelines for entities manufacturing and developing consumer IoT solutions.[13]. This ETSI technical document is based on the UK government's document for the Code of Practice for Consumer IoT Security.[14]

Below we describe the security criteria synthesized for SHEMS from the above sources:

Data Encryption Data should always be encrypted during transport. If sensitive data are stored in the device, they should also be adequately encrypted. When implementing encryption mechanisms, proprietary protocols should be avoided. It should be ensured that SSL/TLS implementations have proper configurations and are up to date [7].

Communication and Connectivity Protection Communication channels between components can be protected by protecting information flow and endpoints. Endpoints have different capabilities and security requirements. This may include mechanisms like network data isolation, network segmentation, firewalls, unidirectional gateways, network access control, etc.[10]

Software/Firmware Security The firmware software is the core of a component. Unauthorized modification of software may result in security threats. Therefore, it should be ensured that software/firmware are protected against unintended and unauthorized updates and modifications. Update Servers (i.e., servers responsible for sending system/firmware updates to the system components) must be trusted and in a secure state so that no illegal software can be sent out as updates. For classical IT systems, examples include the Windows Server responsible for handling updates for other computers in the corporate network, whereas for IoT infrastructures an update server could be responsible for over the air updates like the Zigbee OTA Upgrade Cluster.

Signing update files and validating on the devices before installation may protect illegal installation and updates. If possible, software and firmware should be updated as soon as vulnerabilities are discovered and fixes are available. There should also be the provision to implement scheduled updates. Also, the update process of firmware should take care of unwanted situations like network and power disruptions [22].

Hardware-based Security Controls Hardware protection should go along with the software protection. Software weaknesses and misconfigurations are not the only sources of attacks in the IoT world. One of the factors on which hardware security depends is the security of the micro-controller used in a given device. It also depends on whether a Trusted Platform Module (TPM) is integrated into the component, and how it is used [5, 8]. There are other mechanisms like using Memory Protection Units, incorporating Physically Unclonable Function

[11]Industrial Internet Consortium, "Industrial Internet of Things Volume G4: Security Framework". https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB.pdf

[12]OWASP, "IoT Security Guidance". https://www.owasp.org/index.php/IoT_Security_Guidance

[13]ETSI TS 103 645, https://www.etsi.org/deliver/etsi_ts/103600_-103699/103645/01.01.01_60/ts_103645v010101p.pdf

[14]https://www.gov.uk/government/publications/code-of-practice-for-consumer-iot-security

(PUF), using cryptographic modules, etc., that may contribute to hardware protection of the system [21, 25].

Access Control Access control refers to mechanisms for protecting assets from unauthorized components, based on the business and security requirements (cf. ISO 27001). Access control can be achieved through authentication and authorization mechanisms which validate the interacting components and their privileges against system access.

Cryptographic Techniques There are two types of cryptography namely symmetric and asymmetric cryptography. In symmetric cryptographic techniques, the parties exchanging information share the secret key which is used for encrypting and decrypting messages. Whereas in asymmetric cryptography, one party distributes its public key to other parties who use these to encrypt the message, which can only be decrypted using the private key, which is kept secret. Cryptographic techniques are basically used to ensure confidentiality. These techniques can be implemented for protecting communication and connectivity and establishing secure key management. Examples of its applications useful for IoT and smart grid are message authentication, protected key store, code signing, secure bootstrapping, secure patch management and mutual authentication.[10]

Physical and Environmental Security The system components should be protected against unauthorized physical access. This criterion evaluates how well the system is protected against physical access and environmental conditions. Depending on the context, it may include access control of physical perimeter (area, building, home, room, etc.,) and set of equipment [10]. In the case of equipment, it may have several physical ports accessible that can be misused. Protection mechanisms like disabling unused physical ports or installing equipment with minimal physical ports, physical tamper detection, etc., fall under this criteria [12].

Logging and Monitoring Logging and monitoring help tracking and analyzing activities going on in the system. In case of security incidents, monitored processes and logged data may help to understand the cause and prevent such incidents from happening again. Systems like Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) help to identify and prevent attacks on the system and its components. Thus, evaluating the logging and monitoring mechanisms used by a system is an important security criterion for security classification.

We detail further the security criteria with security functionalities inspired by the IoT Security Compliance Framework proposed by IoT Security Foundation (IoTSF), which is in the form of a checklist [15]. We utilize the security functionality from this framework to fit our need to specify protection levels. Table 6 shows the mapping of security criteria to security functionalities and protection level.

4 Applying the Extended Security Classification to SHEMS

The SHEMS in our case complies with the reference architecture from Section 2.1 and consists of a centralized IoT hub and smart plugs connected to controllable loads such as water heater, air conditioner and floor heating. For simplicity, we do not include storage devices such as batteries that can act as both load and generation device.

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

Table 6: Protection Level Requirements

Protection Criteria	**Security Functionality**	**P5**	**P4**	**P3**	**P2**
Data Encryption	Encryption of data between system components	x	x	x	x
	Strong encryption mechanism	x	x	x	
	Credentials should not be exposed in the network	x	x	x	
	End-to-end encryption	x	x		
	Should not use custom encryption algorithms	x	x		
	Sensitive stored data should be encrypted	x	x		
Communication and Connectivity Protection	Have a minimal number of network ports open	x	x	x	
	Devices should not be accessible from the Internet	x	x	x	
	Only authorized components can join the network	x	x	x	
	Use only standard communication protocol	x	x		
Software /Firmware Security	Updatability of device firmware	x	x		
	Updatability of the operating system	x	x		
	Automatic updates available	x	x		
	Encryption of update files	x	x		
	Signing update files before installing	x	x		
Hardware-based Security Controls	Using Trusted Platform Modules (TPM)	x	x		
	Use of Memory Protection Units (MPUs)	x	x		
	Incorporate Physically Unclonable Functions	x	x		
	Use of Cryptographic Modules	x	x		
Access Control	Disable remote access functionality	x			
	Only authorized devices can join the network	x	x	x	
	Default and weak passwords should not be used	x	x	x	
Cryptography Techniques	Secure bootstrapping	x	x		
	Secure key generation	x	x		
	Secure key storage	x	x		
	Secure key distribution	x	x	x	
	Secure key rotation	x	x		
	Message integrity	x	x	x	
Physical and Environmental Protection	Tamper resistance	x	x		
	Minimal physical ports available	x	x	x	
	Physical security of connections	x	x	x	
	Ability to disable external ports and only minimal ports enabled	x	x		
	Only authorized physical access	x	x	x	
Monitoring and Analysis	Monitoring system components	x	x		
	Analysis of monitored data	x	x		
	Act on analysed data	x			

4.1 Impacts

Below we identify the criticality (**Impacts**) of successful cyberattacks on SHEMS.

Safety. Leakage of data from SHEMS may disclose the presence of people inside their house, which may result in a burglary or other types of crime. Moreover, residents may feel unsafe (reducing trust in SHEMS) if they realize that their privacy is breached and strangers can follow their activities.

Grid imbalance. During the execution of a demand response program, devices that utilize higher energy are turned off to shave the peaks. If an attacker can switch on/off a large number of loads, these may use unexpected amounts of energy that may destabilize the grid [23, 30].

Increased electricity bills. Compromising SHEMS may result in equipment being switched on without authorized persons noticing.

Privacy. Data from SHEMS can be privacy sensitive, as Molina Markham et al. have demonstrated that High-frequency consumption data can be exploited to derive private information such as the number of people in the house, sleep routines, and the presence of babies at home [24]. Compromised SHEMS data may contain even more detailed information. Stealing such data may result in the exposure of personal habits of the residents, which can impact social reputation.

Agents for other cyberattacks. Typically, smart home gateways have connectivity to the Internet. A compromised gateway may act as a bot to launch several other attacks.

Among the aforementioned impacts, grid imbalance and agents for other cyberattacks can be considered as major impacts as these may result in blackouts and damage of physical infrastructures. The remaining impacts could be considered moderate or minor.

4.2 Protection Level for security criteria

Each security criterion demands several security functionalities, adequately configured, in order to reach a given protection level. Here we describe the security functionalities of our system to determine the protection level.

Data Encryption The IoT hub utilizes TLS so that the communication between the IoT hub and the backend system is always encrypted. IoT devices and IoT hubs form a HAN communicating Zigbee. In addition to AES 128-bit encryption, Zigbee also supports end-to-end encryption which is typically safe for mesh networks.

Communication and Connectivity Protection Typical households have a single Local Area Network (LAN). Therefore, if the gateway is connected via Wi-Fi or Ethernet, it forms one of the nodes in the LAN. If an attacker has access to the Wi-Fi network, then the smart home gateway becomes accessible. However, remote access has key-based authentication which reduces the gateway's exposure. Similarly, nodes also have pre-distributed keys, and only authorized nodes can join the radio network.

Software and Firmware Security The capability of a networked system of being upgraded is important for IoT systems. Develco Products smart home system provides the possibility to update the system devices and applications. The operating system is upgraded using SSH which only supports key-based authentication. Data flowing during a system upgrade is also encrypted. Similarly, for firmware upgrades of Zigbee modules, a Zigbee OTA Upgrade Cluster server is implemented. The upgrade process provides security via image verification, authentication, and encryption as specified in Zigbee OTA Upgrade Cluster specification.

Hardware-based Security Controls In this work we have not looked into hardware-based security controls and thus is not applicable.

Access Control Only authenticated IoT devices can join the HAN with the IoT Hub. All devices including the IoT gateway have pre-shared keys which allow them to communicate with each other through encrypted channels. E2U systems use Microsoft IoT hub for communication and management of SHEMS over the cloud. Microsoft IoT hub provides multiple ways to secure communication including token-based and certificate-based authentication mechanisms.[16] Only registered and active IoT hubs(gateways) can communicate with the backend system. End users can only access authorized devices. Administrators have control over the management of smart home gateway systems. Microsoft IoT hub also allows remote monitoring of smart home systems. The security of a smart home system also depends on the security of the backend system. Since we have excluded the backend system and assumed it to be secured, the available access control features are adequate.

Cryptographic techniques The IoT hub and IoT devices are pre-configured and use installation codes, certificates, and pre-defined keys to authenticate the connection. This means that only registered and pre-configured smart home devices can join its assigned gateway. The radio communication uses Zigbee, and data is encrypted using AES 128-bit key. Symmetric link keys (i.e., a unique key for each established link in the network) are used to encrypt data between the gateway and the devices in the network. Similarly, Wi-Fi supports WPA2-PSK (AES/TKIP) encryption. Data integrity is supported by using MIC (Message Integrity Code).

Physical and Environmental Security The system that we have considered has no tamper detection mechanisms for physical security. However, the components of SHEMS are located inside the house and unauthorized users have no physical access to the system.

Monitoring and Analysis The advanced monitoring capabilities include the collection of security data from system components, analysis and taking necessary actions if required based on the analysis. Smart home systems support basic monitoring functionalities where data log information is collected from the devices and sent to the backend. The gateway performs availability checks on its devices and reports to the backend system, but it does not perform extensive security analysis.

Based on our protection level requirements (see Table 6), the protection level of our SHEMS is set to P4.

4.3 Evaluation of Security Class

We limit the presentation of the application of security classification only to Application and Network Data. In particular, we assess the Command and Control (C&C) for a demand re-

[16]Control access to IoT Hub, https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-security.

sponse program, which is one of the most critical components of SHEMS. We apply the classification method for the following two scenarios.

4.3.1 Scenario I: Centralized Control.

In this scenario, Distribution System Operators (DSO) have an agreement with consumers to control the SHEMS appliances to properly manage peaks of energy demand. In our system, each controllable device is plugged into the corresponding smart plug. Depending on the device and their maximum effect, rules for controlling the devices are defined, e.g., a water heater with a maximum capacity 3kW can be controlled only between 8:00 AM to 6:00 PM during weekdays, and once turned off, it cannot be turned on for a minimum of 15 minutes. The DSOs forecast the energy demand in advance and, if reductions are needed, DSOs optimally select the devices to be turned off. Such an operation meets the goal of reducing energy. Control commands are sent to the selected devices from the DSO to meet the goal of targeted reduction of consumption. The state of all affected devices may be reverted back to their original state after the duration of the demand response execution is complete.

Class Evaluation. The connectivity between the IoT device and the hub is C3 (cf. Section 3) and between the hub and the backend system is C5. If an attacker is able to manipulate the device control only inside the HAN (C3), the impact is only Minor. However, if an attacker is able to trigger or manipulate the message for the demand control program from the backend (C5), several devices can be turned off, resulting in grid imbalance as discussed earlier. As a result, for this scenario, we evaluate the overall impact as Major.

To evaluate the security class, we first select the relevant security criteria for C&C as Data Encryption, Communication and Connectivity Protection, Access Control, and Monitoring and Analysis. We then evaluate the protection level based on the strength of the security functionalities in the selected criteria. Using Table 6 and sub-section 4.3, we assign the overall protection level P4.

Using Table 3 we determine from the computed values of connectivity (C5) and protection level (P4), the exposure E3. Using Table 4 we get the class D (Impact Major and Exposure E3), which is a poor score not suitable for SHEMS. To improve the security class, Table 4 indicates that either exposure or impacts need to be reduced. Similarly, exposure can be reduced either by increasing the protection level or by reducing the connectivity, cf. Table 3.

4.3.2 Scenario II: Edge Control.

In this scenario, the control signals are sent by the IoT hub autonomously, based on the time of peak demand or price of electricity, and thresholds set by the end-user. Users can also set priorities for the devices that need to be controlled and rules to decide e.g., when and how long the devices can be controlled. Thresholds and rules can also be persisted in the IoT gateway, allowing control of devices without requiring interaction with the backend.

Class Evaluation. Similarly, if an attacker can manipulate the control message within the HAN network (C3), the impact is considered as Minor. However, since there is no flow of commands from the backend system, an attacker cannot influence many devices on a large scale. Since there are no changes in the protection mechanisms, we can consider it as P4. Moreover, using Table 3(a), we obtain the Exposure E2 and using Table 4 we computed the security class as A.

The analyses of scenario I and II showed that by moving from the centralized control to the edge control for the demand control functionality, the security class of the demand control is significantly improved from class D to class A. Besides, scenario II may even be more efficient and have lower latency because the trigger of device control initiates locally rather than from the backend system to several IoT devices. Such improvements in the design of IoT systems should be considered to improve the security of the overall system.

5 Conclusion and Further Work

In this report, we discuss the architecture of a commercial smart home energy management systems. We also compared three major communication standards for home automation network to discuss their security features. We present the security classification methodology extended with details regarding security functionalities relevant for SHEMS. As an example, we have applied this methodology to the commercial SHEMS from E2U. The example focusses on the C&C part of SHEMS for demand response programs. We have first evaluated the security class of the C&C for a centralized control architecture, resulting in a low and unacceptable security class D. Using our methodology, we can indicate how the system needs to be improved to achieve an acceptable security class. Using an edge controlling concept, our analysis demonstrated an achievable security class A. Further work will focus on aggregation mechanisms for calculating the overall system security class from its components.

References

[1] Giovanni Agosta, Alessio Antonini, Alessandro Barenghi, Dario Galeri, and Gerardo Pelosi. Cyber-security analysis and evaluation for smart home management solutions. In *Security Technology (ICCST), 2015 International Carnahan Conference on*, pages 1–6. IEEE, 2015.

[2] Frances K Aldrich. Smart homes: past, present and future. In *Inside the smart home*, pages 17–39. Springer, 2003.

[3] ZigBee Alliance. Zigbee specification. 2012.

[4] ANSSI. Classification Method and Key Measures. 2014.

[5] Will Arthur and David Challener. *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security.* Apress, 2015.

[6] Ahmad W Atamli and Andrew Martin. Threat-based security analysis for the internet of things. In *2014 International Workshop on Secure Internet of Things*, pages 35–43. IEEE, 2014.

[7] Michael Atighetchi, Nathaniel Soule, Partha Pal, Joseph Loyall, Asher Sinclair, and Robert Grant. Safe configuration of tls connections. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 415–422. IEEE, 2013.

[8] Sergiu Bursuc, Christian Johansen, and Shiwei Xu. Automated Verification of Dynamic Root of Trust Protocols. In Matteo Maffei and Mark Ryan, editors, *6th International Conference on Principles of Security and Trust (POST)*, volume 10204 of *Lecture Notes in Computer Science*, pages 95–116. Springer, 2017.

[9] Daniel Celebucki, Maj Alan Lin, and Scott Graham. A security evaluation of popular internet of things protocols for manufacturers. In *ICCE*, pages 1–6. IEEE, 2018.

[10] Seyed Mahdi Darroudi and Carles Gomez. Bluetooth low energy mesh networks: A survey. *Sensors*, 17(7):1467, 2017.

[11] Bo Fan. Analysis on the security architecture of zigbee based on ieee 802.15. 4. In *Autonomous Decentralized System (ISADS), 2017 IEEE 13th International Symposium on*, pages 241–246. IEEE, 2017.

[12] Xueqi Fan, Fransisca Susan, William Long, and Shangyan Li. Security analysis of zigbee. 2017.

[13] Fadi Farha and Hongsong Chen. Mitigating replay attacks with zigbee solutions. *Network Security*, 2018(1):13–19, 2018.

[14] Puteri Fitriaty, Zhenjiang Shen, and Kenichi Sugihara. How green is your smart house: Looking back to the original concept of the smart house. In *Green City Planning and Practices in Asian Cities*, pages 39–76. Springer, 2018.

[15] Behrang Fouladi and Sahand Ghanoun. Security evaluation of the z-wave wireless protocol. *Black hat USA*, 24:1–2, 2013.

[16] K Ghirardello, C Maple, D Ng, and P Kearney. Cyber security of smart homes: Development of a reference architecture for attack surface analysis. 2018.

[17] James D Gilsinn and Ragnar Schierholz. Security assurance levels: a vector approach to describing security requirements. In *Proceedings of the US DHS industrial control systems joint working group (ICSJWG) 2010 Fall Conference, Seattle, USA*, 2010.

[18] Giwon Kwon, Jeehyeong Kim, Jaewon Noh, and Sunghyun Cho. Bluetooth low energy security vulnerability and improvement method. In *Consumer Electronics-Asia (ICCE-Asia), IEEE International Conference on*, pages 1–4. IEEE, 2016.

[19] Jeong In Lee, Chang-Sic Choi, Wan-Ki Park, Jin-Soo Han, and Il-Woo Lee. A study on the use cases of the smart grid home energy management. In *ICTC*, pages 746–750. IEEE, 2011.

[20] Yuanyuan Liu, Bo Qiu, Xiaodong Fan, Haijing Zhu, and Bochong Han. Review of smart home energy management systems. *Energy Procedia*, 104:504–508, 2016.

[21] Roel Maes and Ingrid Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*, pages 3–37. Springer, 2010.

[22] H. Mansor, K. Markantonakis, R. N. Akram, and K. Mayes. Don't brick your car: Firmware confidentiality and rollback for vehicles. In *2015 10th International Conference on Availability, Reliability and Security*, pages 139–148, Aug 2015.

[23] Amir-Hamed Mohsenian-Rad and Alberto Leon-Garcia. Distributed internet-based load altering attacks against smart power grids. *IEEE Trans. Smart Grid*, 2(4):667–674, 2011.

[24] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 61–66. ACM, 2010.

[25] Thomas Morris. Trusted platform module. In *Encyclopedia of cryptography and security*, pages 1332–1335. Springer, 2011.

[26] John Padgette. Guide to bluetooth security. *NIST Special Publication*, 800:121, 2017.

[27] Felix Redmill. Understanding the use, misuse and abuse of safety integrity levels. In *8th Safety-critical Systems Symposium*, pages 8–10, 2000.

[28] Manish Shrestha, Christian Johansen, and Josef Noll. Criteria for Security Classification of Smart Home Energy Management Systems. In *1st International Conference on Smart Information and Communication Technologies*, Lecture Notes in Electrical Engineering. Springer, 2019. (to appear).

[29] 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 (IJCIP)*, 2019. (to appear).

[30] Saleh Soltan, Prateek Mittal, and H Vincent Poor. Blackiot: Iot botnet of high wattage devices can disrupt the power grid. In *27th USENIX Security Symposium*, pages 15–32, 2018.

[31] Biljana L Risteska Stojkoska and Kire V Trivodaliev. A review of internet of things for smart home: Challenges and solutions. *Journal of Cleaner Production*, 140:1454–1464, 2017.

[32] Bin Zhou, Wentao Li, Ka Wing Chan, Yijia Cao, Yonghong Kuang, Xi Liu, and Xiong Wang. Smart home energy management systems: Concept, configurations, and scheduling strategies. *Renewable and Sustainable Energy Reviews*, 61:30–40, 2016.

[33] Tobias Zillner and S Strobl. Zigbee exploited: The good the bad and the ugly. *Black Hat–2015 https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly-wp.pdf*, 2015.

Paper III

Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT

Manish Shrestha, Christian Johansen, Josef Noll

Published in the Fifth International Conference on Fog and Mobile Edge Computing (FMEC 2020).

DOI: 10.1007/978-3-030-53187-4

A technical report (starting at page 109) supporting this publication was published in January 2020, with the research report number 493, ISBN 978-82-7368-458-5, and ISSN 0806-3036.

The report can be found at:

http://urn.nb.no/URN:NBN:no-80474

III

Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT

Manish Shrestha
eSmart Systems AS
Halden, Norway
manish.shrestha@esmartsystems.com

Christian Johansen
Department of Technology Systems
University of Oslo
Oslo, Norway
cristi@ifi.uio.no

Josef Noll
Department of Technology Systems
University of Oslo
Oslo, Norway
josef.noll@its.uio.no

***Abstract*—The proliferation of IoT (Internet of Things) though making life easier, comes with security and privacy challenges. We have previously proposed a security classification methodology meant to help in practice build IoT systems focused on security during the development process. This method departs from classical risk analysis and certification methods in two ways: (i) it can be used at design time and (ii) it caters for the needs of system designers by helping them to identify protection mechanisms necessary for the connectivity required by their system under development. However, similarly to many risk analysis methods, this methodology was unable to provide assurance in the evaluation results. In this paper, we add two confidence parameters: *belief* and *uncertainty* to the assessment tree of arguments of a class. Thus, the final result is now a tuple <*C, B, U*>, where *C* is the class to which the system belongs, together with a belief measure *B* in the evaluation aspects of *C*, and the uncertainty *U* in the evaluation details. Looking at the confidence parameters tells how well the security assessment is justified. To exemplify this enhanced security classification methodology, we systematically apply it to control mechanisms for Smart Home Energy Management Systems.**

***Index Terms*—Security Classification, Security assurance, Uncertainty, Confidence, Security labelling**

I. INTRODUCTION

Internet of Things (IoT) is widely adopted in major sectors including critical infrastructures such as smart grids and privacy-sensitive domains such as smart homes. Because IoT devices produce sensitive data and have limited memory and processing power, IoT systems are easy targets for launching cyber attacks. Despite the efforts to secure IoT systems, attacks are increasing[1]. One of the reasons behind this is the lack of security awareness in end-users preferring cheaper and easy to install insecure products. Traditional certification approaches s.a. Common Criteria are usually expensive and take more than a year to get certified [2]. Investing in such certification does not pay off because of the lower cost and short life span of IoT products. Even if we start to see standards for cybersecurity for consumer IoT, s.a. the recent ETSI TS 103 645, a framework for designing and evaluating IoT systems for an appropriate level of security still does not exist.

This work is funded by eSmart Systems AS and the Research Council of Norway through the project IoTSec - "Security in IoT for Smart Grids", with number 248113/O70. A long version of this paper is available as [1].

[1]https://www.forbes.com/sites/zakdoffman/2019/09/14/dangerous-cyberattacks-on-iot-devices-up-300-in-2019-now-rampant-report-claims/

We have previously proposed a notion of security classes [3] to address the aforementioned challenges. By systematically applying our security classification methodology, a system designer (or user or certification body, for the same matter) can classify the security of their system on a scale from A to F where A represents the best security level. Most methodologies for security classification or risk analysis are based on knowledge and experience of security experts executing the evaluations. For our target audience, i.e., end-users or system designers/developers, only claiming a security class without justification is insufficient.

To build trust in the security classification one needs to answer questions like: "How confident are the experts in their result?" or "Were any decisions made under uncertainty?". In response, we introduce in this paper two new parameters in the security classification methodology, namely *belief* and *uncertainty*, described in section III after briefly recalling our previous security classification methodology in Section II. In Section IV the enhanced security classification methodology is applied to a Smart Home Energy Management System (SHEMS), ending up with a comparative discussion.

II. SECURITY CLASSIFICATION METHODOLOGY

We have proposed in [3] a security classification methodology, which extends the ANSSI classification, for analysing and evaluating the security of complex connected systems. This methodology is built around three main factors (see Figure 1): Connectivity, Security mechanisms, and Impacts. Connectivity reflects how the system is exposed to attacks, whereas security mechanisms evaluate the security features protecting the system (Protection Level). Connectivity and Protection level combined form the Exposure.

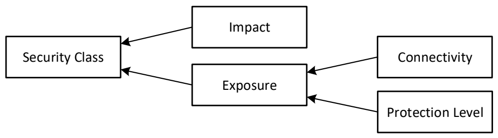


Fig. 1: Process of computing a security class.

We have considered five levels of connectivity (C):

- **C1** : Includes completely closed/isolated systems.
- **C2** : Includes the system with wired Local Area Network and does not permit operations from outside the network.
- **C3** : Includes C2 systems that use wireless technologies.
- **C4** : Includes systems with private or leased infrastructure, which may permit remote operations (e.g., VPN, private APN, etc).
- **C5** : Includes distributed systems with public infrastructure, i.e., like the C4 category only that the communication infrastructure is public.

Similarly, there are five protection levels (P), reflecting the security mechanisms in the system. To determine the protection level, relevant security criteria are defined, and for each criterion, the respective security mechanisms are derived. The security mechanisms are then grouped to form individual protection levels where a higher protection level includes all the security functionalities of lower protection levels, plus additional functionalities. Protection level P1 represents no security mechanisms whereas the protection level P5 represents the strongest protection mechanisms. The evaluation of protection mechanisms is conducted by security experts. Table I guides the evaluation of exposure from connectivity and protection levels.

TABLE I: Calculation 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 II: Calculation 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

The impacts also have five levels taken from ANSSI: Insignificant, Minor, Moderate, Major, and Catastrophic. A security class is determined using impact and exposure according to the look-up Table II.

In a typical SHEMS, devices are remotely controlled (hence, connectivity is C5) and the control data are well encrypted and monitored (hence, protection level P4), and so the acquired Exposure would be E3 (cf. Table I). Given that a compromised SHEMS is seen as a major impact, the final security class would be “D”. Details can be found in [3], [4].

Table II shows variations on these calculations, e.g., exposures E1, E2, or E3 with impact “catastrophic” result in classes A, C, resp. E. However, there are no more explanations than this look-up table, whereas the details of choosing protection mechanisms and connectivity are considered expert judgement art. Hence, we see the need to introduce confidence in the analysis and arguments to justify the results and quantify the (un)certainty with which the decision is made.

III. CONFIDENCE IN A SECURITY CLASS

The main contribution of this paper is to enhance the security classification method with the ability to argue and reflect the level of confidence for each decision. By *confidence*, we mean the degree to which one agrees on the result of the assessment (belief) and the degree to which the expert lacks knowledge about the assessment (uncertainty). To enrich the security classification method, we propose to represent the assessment result using a three tuple *<C, B, U>*, e.g., the evaluation *<A, 84, 16>* means that the result is class A with 84% confidence and 16% uncertainty. The 84% belief is meant to say that we have high confidence in the coverage of all necessary security measures to justify the protection (P), exposure (E), and security class. An uncertainty of 16% would indicate that there is a moderate lack for justification of some of the arguments.

A. Assessment of Belief and Uncertainty

To understand the concept of belief, let us consider a wireless sensor network where an expert makes a claim C1: “Source node adequately encrypts data before sending to the destination”. An expert may justify this claim by referring to the technical documentation from the vendor claiming that data is encrypted during transfer. If the vendor is reliable one may set higher belief on the claim C1, say 90%. However, there may be some errors during design or implementation which may result in unencrypted data. So, the remaining 10% represents the uncertainty of the claim. If one can experimentally verify the C1, e.g. through a penetration testing tool, C1 could be fully trustworthy, which means 100% belief. This 100% is called a plausible belief or plausibility. Hence, plausibility is the maximum belief that can be obtained if all the evidence is provided. In another case where an exploitable flaw is discovered in an encryption algorithm, then disbelief in the claim may arise. Let us say that the estimated disbelief is 30%, then the highest level of belief that one can make in this situation is 70% (reduced plausibility).

One of the widely used approaches to quantifying belief and uncertainty is the Dempster-Shafer theory, which is a generalization of probability theory that allows representing incomplete knowledge by the notion of upper and lower probabilities (belief and plausibility) [5]. Belief (Bel) represents the strength of the existing pieces of evidence that support a given statement. Similarly, Plausibility (Pl) is the upper bound on the belief that could be obtained by adding the evidence to support the statement. Thus, belief is less than or equal to plausibility ($0 \leq Bel \leq Pl \leq 1$).

Uncertainty is the degree of lack of knowledge or evidence to justify the claim. It can be calculated as the difference between plausibility and belief on the acceptance of the claim. Additionally, $Pl < 1$ indicates the existence of disbelief meaning that there is some evidence against the claim. In our context, we reuse the definition of belief and plausibility from the Dempster-Shafer theory. After belief and plausibility evaluation, uncertainty can be calculated as:

$$Uncertainty = Plausibility - Belief \tag{1}$$

B. Specifying security arguments

In a security class evaluation, a series of security arguments are made. Govier defines an argument as "a set of claims in which one or more of them —the premises— are put forward so as to offer reasons for another claim, the conclusion" [6]. To demonstrate an argument in our case, let us take an example of the assessment of the physical security of an IoT device. During the assessment, it was found that the device is located inside the apartment, and is physically accessible only by the residents. Moreover, the device has a tamper detection mechanism which notifies about unauthorized tampering. Therefore, it can be concluded that the IoT device has adequate physical security. In this example, there is a *set of claims and reasons*: IoT device has a secure location because it is installed inside the apartment; IoT device notifies about tamper activities because it has tamper detection mechanism; IoT device has good physical security because it is placed in a safe place and the owner gets notification about device tampering.

The confidence in the conclusion "IoT device has good physical security" depends on the confidence in the reasons presented. The amount of trust in the *ground of the claim* also impacts the confidence. In our example, the ground of the argument is: "if the physical location is secure and a tamper detection functionality exists, the device is physically secure". If the ground is weak, the trust in the conclusion would also be weak. This implies that the amount of confidence in the conclusion depends on the trust in the main claim and the supporting components of the claim.

Properly structured arguments with appropriate justifications and evidence make the expert opinion explicit, resulting in improved communication between experts. This can also help identify missing evidence and poor assumptions. Structured arguments are widely used for justification of decisions, e.g., in safety cases [7], assurance cases [8], [9] and trust cases [10]. In structured cases, the main conclusion is backed up by the evidence. To make the arguments clearer, there are various methods and notations such as Goal Structuring Notation (GSN) [11], Claim Argument Evidence (CAE) [7], or Toulmin argument model [12], that can help experts to structure their arguments. All the above methods represent the argument as a tree structure where the root node is the main claim which further grows into child nodes that provide the justifications using sub-claims and evidence.

Our security classification methodology involves a series of systematic steps to achieve a final security class. We here propose to structure such an assessment as an argumentation model. Structured arguments provide the reasons to support the security claims. These reasons can be seen as security requirements for the assessment, and also guide security experts to determine to which degree the requirements that are fulfilled is reflected by the confidence (belief and uncertainty). Fig. 2 shows an example of the security class evaluation step as an argumentation model using GSN. After the assessment is structured as the argumentation model, the weights, beliefs and plausibility are assigned to the claims and evidence produced.

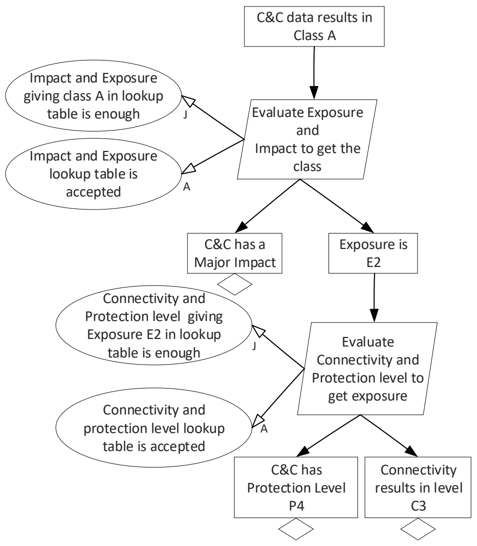

C&C data results in Class A
Impact and Exposure giving class A in lookup table is enough
J
Evaluate Exposure and Impact to get the class
Impact and Exposure lookup table is accepted
A
C&C has a Major Impact
Exposure is E2
Connectivity and Protection level giving Exposure E2 in lookup table is enough
J
Evaluate Connectivity and Protection level to get exposure
Connectivity and protection level lookup table is accepted
A
C&C has Protection Level P4
Connectivity results in level C3

Fig. 2: Class A evaluation using Goal Structuring Notation. (In the figure "J" and "A" point to the justification and assumption made to support the strategy represented by a parallelogram. The rectangle represents the claim and the diamond symbol represents that the digram is incomplete and should be expanded further.)

C. Aggregation of confidence parameters

The result of a security class assessment is represented by the class label with the overall confidence parameter (belief and uncertainty). Thus, after specifying confidence parameters to the security arguments, the result is aggregated to represent an overall assessment. Various aggregation methods have been proposed for structured cases, e.g., Wang et. al. [13] proposed generalized confidence propagation rules for safety cases based on Dempster-Shafer theory. In their D-arg rule, the aggregated belief is calculated as a weighted mean. However, weighted means are not sensitive to extreme lower values of beliefs. Similarly, in their FC-Arg rule, the aggregated belief is the result of multiplying the individual beliefs [13]. The result of such beliefs would always be diminished if we add more evidence that has belief less than 1. However, normal intuition is that, with an increase of evidence, the beliefs should strengthen. Aggregation rules in trust cases have similar problems, e.g., see aggregation rules C-arg, SC-arg and NSC-arg in [10]. Noll et. al., in their Multi-Metrics (MM) approach [14], have claimed that quadratic functions reflect the aggregation better than linear approaches.

In our case, the arguments we have considered contribute individually to the overall goal. Based on the significance of each component in the system's security, we assign appropriate weights in the range [0-100]. We then compare the weighted mean approach with MM approach for calculating beliefs.

1) Weighted mean approach: The aggregated belief using weighted mean for beliefs (b) and weights (w) can be calculated using the formula:

$$AggregatedBelief(c) = \frac{\sum_i^n w_i b_i}{\sum_i^n w_i} \tag{2}$$

2) MM approach: The MM approach uses the Root Mean Squared Weighted Data (RMSWD) to aggregate criticality values and is expressed as:

$$X = \sqrt{\sum_i \left(\frac{x_i^2 W_i}{\sum_i^n W_i} \right)} \tag{3}$$

where X is the aggregated criticality, x_i is the criticality of i^{th} component, and W_i is calculated from the component weight w_i as:

$$W_i = \left(\frac{w_i}{100} \right)^2 \tag{4}$$

In the original work, criticality x_i is defined as the complement of security, privacy or dependability metrics [14]. In our context, we use the complement of belief value (*100 - belief*) to express criticality. Finally, the aggregated belief (Bel) is computed as a complement of X (i.e., $Bel = 100 - X$). Thus, using equation 3, Bel can be expressed as:

$$Bel = 100 - \sqrt{\sum_i \left(\frac{(100 - bel_i)^2 W_i}{\sum_i^n W_i} \right)} \tag{5}$$

where bel_i is the individual belief value of the component under consideration.

D. Underlying principles for aggregation

Belief aggregation depends on how the arguments are presented. There are cases when there are multiple justifications independent of each other fulfilling the same claim, or sometimes each justification contributes towards the fulfillment of the claim to some extent. Here we describe the principles to guide the aggregation mechanism in special cases.

1) **Maximum belief**: If justifications are overlapping and one justification includes another, the highest belief should be considered. For example, to justify the claim "Data is encrypted", there are two evidences with different beliefs: 1) Document from the vendor describing that the data is encrypted (belief = 90%), and 2) Experimental verification for encryption (belief = 100%). In this case, we simply select the highest belief because the information from the vendor's document is subsumed by the experimental verification. Thus,

   $Aggregated\ Belief = Max(b1, b2)$

   where $b1$ and $b2$ are beliefs on overlapping claims where justification of one of the claim includes the other.
2) **Zero belief**: If the belief for any of the claims in the evaluation for protection level is zero, then the total belief should be zero because the class is determined based on the previously specified requirements of security functionality (*sf*). If one of the functionality has no belief at all, then the whole claim for that protection level fails and it must be evaluated against the lower protection level. The same applies to the aggregation of protection criteria (c) towards the protection level.

```
if $c.securityFunctionalities.Any(sf.belief = 0)$ then
    $c.belief = 0$
end if
```

3) **Minimum belief**: Typically, it is assumed that the impacts and connectivity have full beliefs; otherwise, if the beliefs are lower than 100%, the resulting aggregated belief should be the lowest one. For example, if the exposure is E2, with belief 90% and the Impact is Major with 60% then the class obtained should have belief 60% instead of the average. This is because both of them are equally important and required for evaluation. Thus, the averaged belief has no meaning. Hence,

   $Aggregated\ Belief = Min(b1, b2).$

IV. CASE STUDY

To demonstrate the applicability of confidence in security classifications, we have selected a use case involving command and control in SHEMS. The scenarios for the use case are built upon our previous work [4] which used two principal methods to control the IoT devices: centralized and edge control. The centralized control has higher connectivity and major impacts, therefore resulting in class D; whereas, in the edge control scenario, the connectivity is reduced, while also reducing the impacts, thus resulting in class A. We continue here to look at the edge control scenario and follow Section III to add confidence reasoning.

A. Protection level evaluation

For an IoT device control system, we considered Data Encryption (e.g., for securing control commands), Access control, and Monitoring & Analysis, as relevant criteria. We first analyze the security mechanisms available for each of these security criteria, in order to determine the protection level following the summary in Table III. We assume that the answers for the existing security functionalities in the C&C component fall onto the column (i.e., protection level) P4, i.e., two functionalities are not present. Next, we discuss confidence parameters for each protection criteria and their mechanisms.

1) Data Encryption: The following sub-claims were considered to satisfy the P4 level requirements:

- **C&C data is encrypted between IoT hub and devices**: The belief on this claim is 100% and is justified by the vendor's document and lab test.
- **Data encryption uses a strong encryption algorithm**: It has been verified that data is encrypted with AES 128-bit encryption which is considered strong for home network. Thus, the belief in this sub-claim is also set to 100%.

TABLE III: Protection Level Requirements for C&C in a SHEMS application.

Protection Criteria	Security Functionality	P5	P4	P3	P2	Table IV
Data Encryption	C&C data is encrypted between IoT hub and devices	✓	✓	✓	✓	✓
	Data encryption uses a strong encryption algorithm	✓	✓	✓		✓
	End-to-end encryption is supported	✓	✓			✓
	Does not use custom encryption algorithms	✓	✓			✓
Access Control	Disable remote access functionality	✓				
	Weak and default credentials are not allowed	✓	✓	✓		✓
	Enable Multi-factor Authentication	✓	✓			N/A
Monitoring and Analysis	Monitor system components	✓	✓			✓
	Analysis of monitored data	✓	✓			✓
	Act on analysed data	✓				

- **End-to-end encryption is supported**: Communication uses Zigbee which supports end-to-end encryption. However, we did not find any claims from the vendor about end-to-end encryption. We also did not experimentally verify this claim. Thus, this claim is partially trusted (50%) but has the plausibility of 100% if verified experimentally or claimed by the vendor.
- **Does not use custom encryption algorithms**: This sub-claim has 100% belief because the communication uses the Zigbee protocol with a standard AES 128-bit encryption.

2) Access Control: In our case, a C&C command is triggered based on a predefined threshold setting. The C&C command is sent from the IoT hub to the devices in the home network. We consider the following sub-claims to fulfil P4:

- **Weak and default credentials are not allowed**: The hub and the devices are authenticated using pre-shared unique keys allowing only authorized nodes access to C&C data. The C&C data has restrictions to be accessed and triggered only by the gateway. Thus, we consider access control as adequate and assign the belief of 100%.
- **Enable Multi-factor Authentication**: This claim is not relevant because the control signals are sent autonomously and thus user authentication is not involved.

3) Monitoring and analysis: The claim for this criterion can be supported by the following two sub-claims:

- **C&C data is adequately monitored**: The SHEMS in our context supports basic monitoring. The log information such as devices status and control signals are collected. Thus, the assigned belief is 98% because we have not done testing in the lab of the logging system for bugs.
- **C&C data is adequately analyzed**: The gateway performs regular availability check on its devices and notifies about the disconnection of device(s). Though it is possible to manually analyze the monitored data more thoroughly from the log, such more extensive security analysis on collected data is not performed. Thus, the sub-claim has a lower belief set to 80%.

Table IV summarizes the beliefs, plausibilities and weights (w) assigned to the parameters for protection level evaluation of the selected criteria.

B. Aggregation using the weighted mean approach

Since there was no disbelief, and we calculate

$$Plausibility = 1 - Disbelief \tag{6}$$

then the plausibility in all cases was 100%.

Using the weighted mean approach (Equation 2), we calculated the aggregated belief for Data Encryption criterion as 89%, Access Control as 100% and Monitoring & Analysis as 89%. Further aggregation gave us 93% belief on the claim of protection level P4. Thus, we assign the overall confidence to the class A evaluation as 93% belief and 7% uncertainty, i.e. <A, 93, 7>.

C. Aggregation using MM approach

Using this approach, the aggregated belief for Data encryption, Access Control and Monitoring & Analysis criteria obtained were 78%, 100% and 86%. Similarly, the aggregated belief for P4 was 84%. Since plausibility was considered 100% all the time, it does not change after aggregation. The resulting class obtained was class A with 84% belief and 16% uncertainty i.e., <A, 84, 16>. Table V summarizes the results from the weighted mean and MM approach.

V. Analysis and Discussions

We compared two approaches to aggregate beliefs. The weighted mean approach is not sensitive to low values. For instance, among the data encryption criteria from Table IV, there is one security functionality with weight 80 and belief 50. However, the aggregated belief is calculated as 89% in Table V. This value is not very realistic, because in security if one of the claims has low belief, it may have a high effect in the overall security (i.e., the "weakest-link" principle). Hence, the lower values should be well reflected in the aggregation of beliefs in security. When applying the MM approach to aggregate the belief for the same criterion (Data Encryption), we obtain the aggregated value of 78%, which is somewhat more realistic than 89%; suggesting the MM approach to aggregation of beliefs as preferable.

Because there is no disbelief in our case, the plausibility is 100%. Thus, the overall evaluation using the MM approach produced a belief of 84% and an uncertainty measure of 16%. The uncertainty can be reduced by providing missing evidence, e.g., the belief in the existence of end-to-end encryption can be increased by performing experimental validation.

TABLE IV: Belief, Plausibility and Weights on security claims.

Protection Criteria	**Security Functionality**	**<Bel, Pl, w>**
Data Encryption (w=100)	C&C data is encrypted between IoT hub and devices	<100, 100, 100>
	Data encryption uses a strong encryption algorithm	<100, 100, 95>
	End-to-end encryption is supported	<50, 100, 80>
	Does not use custom encryption algorithms	<100, 100, 95>
Access Control (w=95)	Weak and default credentials are not allowed	<100, 100, 100>
Monitoring and Analysis (w=80)	Monitor system components	<98, 100, 100>
	Analysis of monitored data	<80, 100, 95>

TABLE V: Comparison of weighted mean and Multi-Metrics approach for belief aggregation.

Weighted Mean Approach		**Protection Criteria**	**Multi-Metrics Approach**	
Aggregated to Protection Level	**Aggregated to Criterion Level**		**Aggregated to Criterion Level**	**Aggregated to Protection Level**

Bel = 93%
Pl = 100%
U = 7%
	Bel = 89%
	Pl = 100%
	U = 11%
		Data Encryption (w=100)	Bel = 78%
			Pl = 100%
			U = 22%%
				Bel = 84%
				Pl = 100%
				U = 16%

	Bel = 100%
	Pl = 100%
	U = 0%
		Access Control (w=95)	Bel = 100%
			Pl = 100%
			U = 0%	

	Bel = 89%
	Pl = 100%
	U = 11%
		Monitoring & Analysis (w=80)	Bel = 86%
			Pl = 100%
			U = 14%	

Both uncertainty and disbelief increase unreliability. However, higher disbelief shows unreliability with certainty (obtained via evidence) indicating a weaker statement. As an example of disbelief, let us say that a claim is made for adequate data encryption for Wi-Fi communication using WEP standard. The disbelief in the claim is high and the uncertainty is low because, although WEP provides encryption, it is proven to be weak. Hence, to reduce the disbelief, the WEP must be upgraded to a more secured standard such as WPA2.

In the assessment, if the belief is too low, and the uncertainty is high, then the assessment requires more work or the experts may have less knowledge about the security built into the system. However, if the belief is low and the disbelief is high, it means the claims made in the assessment are not trustworthy. Therefore, appropriate measures should be taken to improve confidence. Similarly, an acceptable but not too high value of beliefs may say that the claims are trustworthy but not fully acceptable. In terms of security class, it may mean a different level of trust in the assessment. For instance, a claim of Class A with belief 60% and plausibility 95%, may mean a less trusted class A (which we could denote as A^{--}), while a belief of 95% may represent a highly trusted class A (e.g. A^{++}).

VI. Conclusion

We have shown how to extend security classifications with confidence parameters (i.e., belief and uncertainty) focusing on the methodology presented in [3]. We then exemplified the calculation of confidence parameters for a use case involving an edge command & control mechanism for SHEMS. We compared two types of methods for aggregating confidence measures, observing that weighted average methods are less suitable for security assurance than methods based on Root Mean Squared Weighted Data as the one used to aggregate “criticality” in the Multi-Metrics method of [14]. There are though different principles guiding when to consider minimum, maximum, or zero belief during aggregation.

Further work is needed for building these argumentation and aggregation methods into a tool, following works on tools like NOR-STA [10]. This work is more difficult to integrate with the security classification methodology that comes with predefined mechanisms and look-up tables.

References

[1] Manish Shrestha, Christian Johansen, and Josef Noll. Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT (long version). Technical Report 493, Uni. Oslo, January 2020.

[2] Gianmarco Baldini, Antonio Skarmeta, Elizabeta Fourneret, Ricardo Neisse, Bruno Legeard, and Franck Le Gall. Security certification and labelling in internet of things. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 627–632. IEEE, 2016.

[3] 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*, 2020.

[4] 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.

[5] Jean Gordon and Edward H Shortliffe. The dempster-shafer theory of evidence. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, 3:832–838, 1984.

[6] Trudy Govier. *A practical study of argument*. Cengage Learning, 2013.

[7] Peter Bishop, Robin Bloomfield, and Sofia Guerra. The future of goal-based assurance cases. In *Assurance Cases*, pages 390–395, 2004.

[8] Tim Kelly. Reviewing assurance arguments-a step-by-step approach. In *Workshop on assurance cases for security-the metrics challenge, dependable systems and networks (DSN)*, 2007.

[9] John Goodenough, Howard Lipson, and Chuck Weinstock. Arguing security-creating security assurance cases. *rapport en ligne (initiative build security-in du US CERT), Université Carnegie Mellon*, 2007.

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

[11] John Spriggs. *GSN – The Goal Structuring Notation: A Structured Approach to Presenting Arguments*. Springer, 2012.

[12] Stephen E. Toulmin. *The uses of argument*. Cambridge Uni.Press, 2003.

[13] Rui Wang, Jérémie Guiochet, Gilles Motet, and Walter Schön. Safety case confidence propagation based on dempster–shafer theory. *International Journal of Approximate Reasoning*, 107:46–64, 2019.

[14] Josef Noll, Inaki Garitano, Seraj Fayyad, and Habtamu Abie. Measurable security, privacy and dependability in smart grids. *Journal of Cyber Security and Mobility*, 3(4):371–398, 2014.

UiO : Department of Informatics
University of Oslo

Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT (long version)

Manish Shrestha , Christian Johansen , Josef Noll

Research report 493, January 2020

ISBN 978-82-7368-458-5
ISSN 0806-3036

Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT (long version)

Manish Shrestha Christian Johansen Josef Noll

January 2020

Abstract

We have previously proposed a security classification methodology for IoT systems and have applied it to the smart grid and smart home domain. This method departs from classical risk analysis and certification methods in that it caters for security at design time and for the system designers' needs to know what protection mechanisms to use for the connectivity and exposure that their system under development requires. Though this method can be used for certification, after the system was built, much of the benefit comes in using it to decide what security features to choose to reach the desired security class. However, similarly to many risk analysis methods, this methodology is unable to assure the evaluation results by properly justifying the assessment. In this work we add two confidence parameters: *belief* and *uncertainty* to the assessment tree of arguments of a class. Thus, the final result will now be a tuple $<C, B, U>$, where C is the class to which the system under consideration belongs, along with a belief measure B in the evaluation aspects of C, and the uncertainty U in the evaluation details. Looking at the confidence parameters tells how well the security assessment is justified. To exemplify this enhanced security classification methodology, we systematically apply it to two control mechanisms for a Smart Home Energy Management Systems.

[0]This is an extended version of the conference paper published as [18].

[0]*Address for correspondence:*
Department of Technology Systems, University of Oslo, P.O. Box 70, 2027 Kjeller, Norway.
E-mail: `cristi@ifi.uio.no`

Contents

1 Introduction

When the Internet of Things (IoT) emerged, it was designed for purely functional purposes to provide sensing, connectivity, and controlling features at a lower cost without focusing on security and privacy. Today, IoT is widely adopted in major sectors, including critical infrastructures such as smart grids and privacy-sensitive domains such as smart homes. Because IoT devices produce valuable data and have limited memory and processing power, IoT systems are easy targets for launching cyber attacks. Since the number of attacks is only increasing, security and privacy is a major concern[1].

The majority of end-users of IoT devices, such as elderly living in smart homes providing automated and at distance health-care for their residents, lack security awareness. Some may have a lack of technical understanding to select a secure system from the full range of products available on the market. Many consumers who have little or no understanding of the value of having secure products may prefer cheaper and easy to install products with no security concerns. Moreover, to get a competitive advantage, manufacturers also focus on producing devices at lower costs. A more comfortable option to cut down the manufacturing cost is to compromise the non-functional requirements such as security, and maintain the functionality. Thus, system designers and developers lack incentives to build secure systems.

There is also a lack of regulations and standards that can be enforced towards the manufacturers and vendors to produce secure IoT products and services. The traditional certification approach, such as Common Criteria, is highly expensive and takes quite a long time to get certified [2]. Investment in such certification is not worthy because of the lower cost and shorter life span of IoT products. By the time when one product is certified, it may already be too old in the market. Thus, we need an affordable framework that can guide to secure not only an IoT device but the IoT system as a whole, where all parties, including end-users, manufacturers & vendors, and regulatory bodies, can together contribute towards the secure IoT systems.

European Telecommunications Standards Institute (ETSI) has published the technical specification ETSI TS 103 645, entitled "Cybersecurity for Consumer Internet of Things" as a standard that provides thirteen guiding principles to the stakeholders involved in development and manufacture of consumer IoT to secure their products.[2] ETSI TS 103 645 is based on the "Code of Practice for Consumer IoT Security" published by the Government of UK.[3] Providing general guidelines towards all consumer IoT devices, ETSI TS 103 645 is a good starting point for compliance with the standard. However, based on the severity and criticality of each specific use case of an IoT system, the level of security required may vary, and thus it is essential to have a framework focused on security levels and their overall requirements and thus guidelines.

To address the aforementioned challenges, we have proposed a notion of security classes [19]. Using security classes, one can describe the system in terms of network connectivity, security mechanisms, and the degree of consequences of a security breach (also called impact). By systematically applying this security classification methodology, one can label the security of a system from A to F, where A represents the best security level, whereas F represents the worse security level. This methodology can be used to specify the security requirements and perform the security assessment for the targeted class. While a security label can increase the security awareness of end-users, regulatory bodies can utilize the classes to enforce the vendors and manufacturers to produce the systems aligned with the specified security class.

In standard methods such as risk assessment and certification programs in security, the security assessment and most of the decisions are based on the knowledge and experience of

[1]https://www.forbes.com/sites/zakdoffman/2019/09/14/dangerous-cyberattacks-on-iot-devices-up-300-in-2019-now-rampant-report-claims/

[2]https://www.etsi.org/deliver/etsi_ts/103600_103699/103645/01.01.01_60/ts_103645v010101p.pdf

[3]https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/773867/-Code_of_Practice_for_Consumer_IoT_Security_October_2018.pdf

security experts. However, a making claim is not enough in the expert-based evaluation. How confident are the experts in their results? Did they make any decisions under uncertainty? These questions need to be answered to build trust in the security class methodology. Thus, in this technical report, we investigate to answer these questions by introducing two new parameters, namely belief and uncertainty. Section 2 briefly recalls our security classification methodology. Section 5 describes the concept of belief and uncertainty. Section 6 presents the enhanced security classification methodology using a case study of a home energy management system. Section 7 discusses the results of the use case, and finally, section 9 concludes the report.

2 Security classification methodology

In [19] we have proposed a methodology for analyzing and evaluating the security of complex connected systems. This methodology is built around three main factors: connectivity, security mechanisms, and impacts. On one hand, connectivity reflects how the system is exposed to attacks. On the other hand, the protection level reflects what security features/mechanisms are built into the system and what security functionalities are available. Connectivity and protection level combined form the exposure. Figure 1 visually describes the security classification methodology.

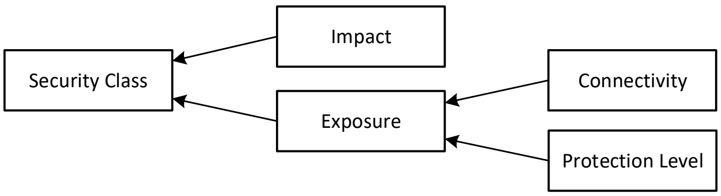


Figure 1: Methodology for computing a security class (arrows indicate that a component contributes to the component that it points to).

In the classification method, we have considered five levels of connectivity (C):

- **C1**: Includes completely closed/isolated systems.
- **C2**: Includes the system with wired Local Area Network (LAN) 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.

Similarly, there are five protection levels (P) that reflect the strength of the security mechanisms in the system. The protection level increases with the increasing number of security mechanisms. Relevant security criteria should be defined to determine the protection level. Then for each security criterion, several security mechanisms are derived. The security mechanisms are then grouped to form individual protection levels where higher protection levels include all the security functionalities of lower protection levels, including some additional functionalities. Protection level P1 represents no security mechanisms, whereas the protection level P5 represents the most robust protection mechanisms. Security experts specify the levels of protection mechanisms. Exposure is then evaluated using protection level and connectivity. Table 1(a) shows the evaluation of exposure from connectivity and protection level.

Table 1: Calculations of: (a) exposure levels and (b) security classes.

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

The impacts, or the consequences, also have five levels: Insignificant, Minor, Moderate, Major, and Catastrophic. A security class is determined using impact and exposure. Table1(b) shows the lookup table for identifying the class from exposure and impact levels.

In a typical home energy management system, the devices can be remotely controlled (i.e., making Connectivity C5), and the control data are well encrypted and monitored (i.e., typically reaching a protection level P4), and thus the acquired exposure is E3 (cf. Table 1(a)). Given that a compromised home energy control system is seen as having a major impact, the resulting security class would be "D". Details of security classification methodology can be found in [17, 19]. Table 1(b) shows the minor variations, e.g., exposure E1, E2, and E3 with impact "catastrophic" results in security classes A, C, and E.

In the security classification method, the majority of decisions are based on the subjective judgment of security experts. Thus, it is vital to provide explicit explanations of why each decision was made. Besides, experts may not always have full confidence in their decisions. Hence, we need to introduce confidence in the security classes' analysis to justify the results and make sure that the decisions are made with certainty. In the next section, we discuss introducing two new parameters to security class, belief, and uncertainty.

3 Confidence in a security classification

Representing the security analysis only through a class gives an idea about the security features and connectivity of a system. However, since security classifications are done by human experts, be that specialised security experts from outside the company (like a penetration team or certification company) or from inside the company (like a CSO), we need the ability to evaluate the amount of confidence we have in the respective classification. This is especially so when the classification is done by not-so experts, like inside a company, in a DevOps process, like our classification is meant for. By confidence, we mean the degree to which one agrees on the result of the assessment (belief) and the degree to which the expert lacks knowledge about the assessment(uncertainty).

We enrich the security classification method by representing the assessment result using three-tuples $< Class, Belief, Uncertainty >$. Fr example, in the use case from this work, the resulting class was A with 84% confidence and 16% uncertainty, represented as $< A, 84, 16 >$. In this assessment, 84% belief would mean that we have high confidence in the coverage of all the security measures being necessary to justify the protection (P), exposure (E), and security class

(S). Uncertainty of 16% indicates a moderate lack of justification for some of the arguments concerning P, E, and S.

The confidence parameters of the class depend on all the lower-level decisions made to compute a class. For example, confidence in class depends on the confidence in impact and exposure evaluation. Confidence of exposure depends on the confidence in connectivity and protection level evaluation and so on. Thus, to compute the overall confidence over the decision, confidence over each smaller evaluation should be considered.

Adequate arguments to support the decisions should be provided, and the confidence parameters should be assigned to each argument. To assign the confidence parameters, one must justify through a proper explanation, why a particular value of confidence is assigned for a given evaluation.

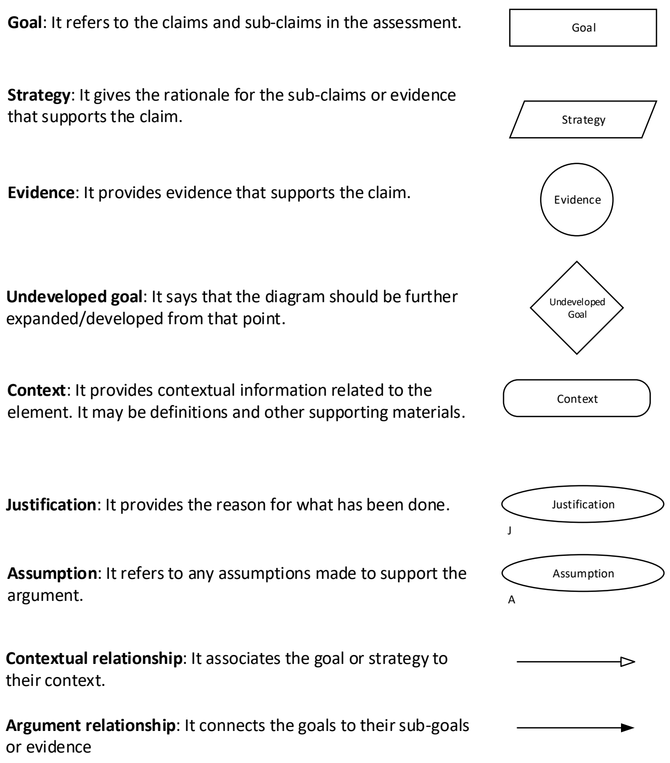
4 Specifying security arguments

One useful way of viewing the security classification method is to specify the security guideline for a targeted class. The methodology can also be used to evaluate a given system against a given class. The assessment is typically based on verification of whether the given requirements for a specified class are fulfilled. Since the assessment is subjective, for the trustworthiness of the assessment it is crucial to justify the result of the assessment. Justification can be achieved by producing relevant arguments supporting each decision taken in the assessment. Govier defines an argument as "a set of claims in which one or more of them-the premises- are put forward so as to offer reasons for another claim, the conclusion" [10].

An argument typically consists of the main claim (conclusion), supported further by the evidence or sub-claims to justify its parent claim. Well-structured arguments make the expert opinion explicit resulting in better documentation of claims and improved communication between the experts. It can also help to identify missing evidence and weak assumptions made during the evaluation. Structured arguments are widely used in assurance cases [4, 11]. An assurance case is similar to a legal case where arguments are presented to support the claims, backed by evidence [8].

Security class evaluation involves a series of security arguments. To demonstrate how an argument is formed, let us take an example of an assessment of an IoT device's physical security. It is considered that for an IoT device to be physically secured, two requirements should be fulfilled. First, the device is placed in a secured area, and second, if somebody tries to fiddle with the device, the owner should get a notification about it. During the assessment, it was found that the device is typically located inside the apartment. Thus, only people living in the apartment have physical access to the IoT device. Also, the device has a tamper detection mechanism that notifies the owner when someone tampers with the device. Therefore, it was concluded that the IoT device has adequate physical security. In this example, there are a set of claims and reasons: IoT device is located in a safe place because it is located in the apartment; IoT device gets a notification about tampered device because it has tamper detection mechanism; IoT device has good physical security because it is placed in a safe place and the owner gets a notification about device tampering. The confidence in the main conclusion "IoT device has good physical security" depends on the confidence in the reasons presented. The amount of trust in the ground of the claim also impacts confidence. In our example, the ground of the argument is, if the physical location is secure and the tamper detection functionality is secure, the device is physically secure. If the ground is weak, the trust in the conclusion would also be weak. It implies that the level of confidence in the conclusion depends on the trust in the main claim and also on the supporting components of the claim. Thus, to properly evaluate the confidence in the decisions, the security arguments must be properly structured so that the overall confidence can be evaluated by utilizing all the related claims.

Representation of arguments using only text as above lacks readability and maintainability. There are various diagrammatic notations such as Goal Structuring Notation (GSN) [20], Claim Argument Evidence (CAE) [4], or Toulmin argument model [21], that can help experts to structure their arguments. All the above methods structure the argument in a tree structure where the root node is the central claim, which further grows into child nodes that provide the justifications using sub-claims and evidence. These methods are often used in constructing assurance cases. There are also several tool support implementations using a structured argument approach to construct assurance cases [12], where most of them comply with the GSN standard.



Goal: It refers to the claims and sub-claims in the assessment.

Strategy: It gives the rationale for the sub-claims or evidence that supports the claim.

Evidence: It provides evidence that supports the claim.

Undeveloped goal: It says that the diagram should be further expanded/developed from that point.

Context: It provides contextual information related to the element. It may be definitions and other supporting materials.

Justification: It provides the reason for what has been done.

Assumption: It refers to any assumptions made to support the argument.

Contextual relationship: It associates the goal or strategy to their context.

Argument relationship: It connects the goals to their sub-goals or evidence

Figure 2: Summary of Goal Structuring Notation (GSN) symbols.

The security classification process involves a series of systematic steps to achieve a security class. We propose to assess the confidence in a system's security class by using a structured argumentation model. Structured arguments provide the reasons to support the security claims. These reasons can be seen as security requirements, and to which degree these requirements are fulfilled is reflected by the confidence (belief and uncertainty). In this particular work, we use GSN to represent our argumentation model diagrammatically. We choose GSN because it is simple, popular, and has open-source tool support to construct the argumentation model [12] [4,5,6]. GSN provides a set of symbols to represent the argument, summarized in Figure 2. Using

[4]Certware, https://nasa.github.io/CertWare/

[5]Acedit, https://github.com/arapost/acedit

[6]D-case editor, https://github.com/d-case/d-case_editor

the GSN diagram, Figure 3 shows the security class evaluation step as a GSN argumentation model.

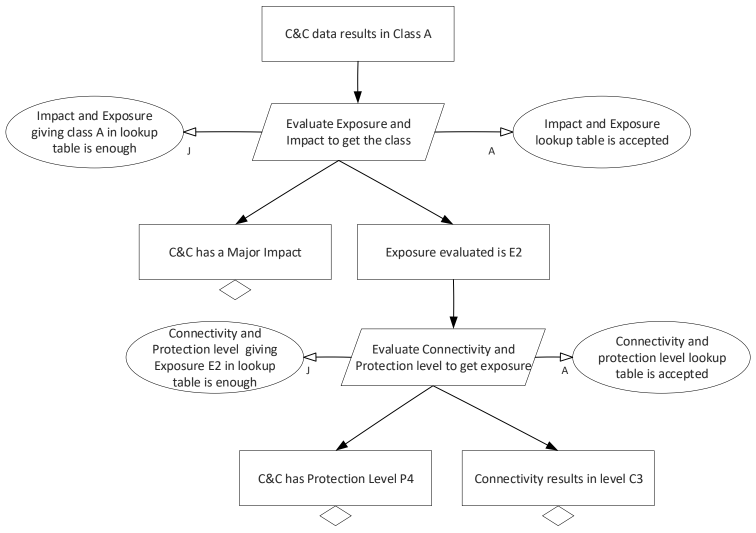

C&C data results in Class A
Impact and Exposure giving class A in lookup table is enough
J
Evaluate Exposure and Impact to get the class
A
Impact and Exposure lookup table is accepted
C&C has a Major Impact
Exposure evaluated is E2
Connectivity and Protection level giving Exposure E2 in lookup table is enough
J
Evaluate Connectivity and Protection level to get exposure
A
Connectivity and protection level lookup table is accepted
C&C has Protection Level P4
Connectivity results in level C3

Figure 3: Representing class A evaluation using Goal Structuring Notation (GSN).

5 Assessment of belief and uncertainty

The degree of confidence in the justification can be reflected using two parameters, belief and uncertainty. To understand the concept of belief, let us consider a wireless sensor network where an expert claims Cl1:"Source node adequately encrypts the data before sending it to the destination node". We need a basis to trust this claim. Since the vendor is popular in the market, and many organizations trust them, most people will also trust the vendor. The vendor also provides a document stating that all nodes encrypt data before the transfer, and explains how it is done at a very high level. With all these, one may set the belief on the claim Cl1 at 90%. However, during the design or implementation, there may be errors/bugs resulting in data being unencrypted. Thus, the remaining 10% represents the uncertainty of the claim.

Now, if we can experimentally verify that sensor data are encrypted, e.g., through a penetration testing tool, we can completely trust the claim, which means 100% belief. However, we do not have the equipment or knowledge to do that, and thus, the claim is not yet verified. Thus, this 100% is called the plausible belief or plausibility. Plausibility is the maximum belief that can be obtained if all the evidence can be provided. Sometimes, plausibility cannot reach 100%. For instance, if a flaw is discovered in the encryption algorithm allowing unauthorized decryption, then the disbelief in the claim may arise. Let us say that disbelief is estimated at 30%. That means the highest level of belief that one can make in this situation is 70%.

One of the widely used approaches to quantify the belief and uncertainty is Dempster-Shafer theory [6, 16]. It is a generalization of probability theory that allows for incomplete knowledge by offering the notion of upper and lower probabilities called belief and plausibility functions [9]. Belief shows the strength of the existing evidence supporting a given statement and represents the lower bound probability of validating the statement. Plausibility is the upper bound of the belief that could be obtained by adding the evidence to support the statement or claim.

To formally describe Dempster-Shafer's belief and plausibility, let us consider a frame of discernment X (i.e., the set of all possible states of system under consideration) and 2^X be power set of X. The belief mass, or the basic belief assignment, m can be assigned to each element of 2^X where $m : 2^X \to [0, 1]$. If A is an element of 2^X, then:

$$\sum_{A \subseteq 2^X} m(A) = 1 \tag{1}$$

The belief and plausibility function for A can be defined as:

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{2}$$

$$Pl(A) = \sum_{B \cap A \neq \phi} m(B) \tag{3}$$

Thus, belief $\leq$ plausibility. Both belief and plausibility have the range [0,1].

By definition, uncertainty is the degree of lack of knowledge or evidence to justify the claim. Thus, the difference between plausibility and belief reflects the uncertainty on the acceptance of the claim. Additionally, the plausibility value less than one indicates the existence of disbelief, meaning that there is some evidence against the claim.

$$Plausibility = 1 - Disbelief \tag{4}$$

Equation 4 shows that in the absence of disbelief, plausibility is 100%. In our context, we reuse the definition of belief and plausibility from the Dempster-Shafer theory. Evaluating plausibility is more natural than to evaluate uncertainty in the first place. Thus, we evaluate the belief and plausibility first and calculate uncertainty as:

$$Uncertainty(u) = Plausibility(Pl) - Belief(b) \tag{5}$$

Security class assessment involves a single conclusion (claim for a given class) decomposed into claims and sub-claims. During the assessment, the confidence is assigned to each of the (sub)claims. The confidence may be assigned based on the evidence presented to support the claims. It is essential to combine all the confidence to produce an aggregate (concluding confidence) that represents the overall confidence of the assessment.

5.1 Aggregation of confidence parameters

In argument based approaches such as in safety cases, which are based on the Dempster-Shafer approach, the belief aggregation rules defined are based on the type of arguments. Each claim typically has weights that represent its significance. For example, in trust cases, Cyra and Górski [5] proposed the aggregation rules based on four types of arguments Complementary (C) arguments, Necessary and Sufficient (NSC) arguments, Sufficient (S) arguments, and Alternative (A) arguments. In their Visual Assessment of Argument (VAA) approach, they define linguistic confidence and decision scales for user's ease. During the evaluation, the linguistic scale is converted to belief and plausibility values for aggregation. After aggregation, the belief and plausibility values are again converted back to confidence and decision values to show in the linguistic scale. They use scaling functions for mapping the linguistic confidence and decision scale into the [0-1] interval. However, the values of the parameters of the scaling functions are not defined. Also, the aggregation rules defined may not fit in the security domain. For instance, the aggregation rule for C argument is a weighted mean of beliefs of child components of a claim. However, weighted means are not sensitive to lower values of beliefs. Also, the

NSC and SC arguments aggregate the value by multiplying the individual beliefs. The aggregated result of such beliefs always diminishes if we add more evidence that has a belief less than 1. However, the rational intuition is that, with an increase of evidence, the beliefs should strengthen.

Similarly, Wang et al. [23] have proposed generalized aggregation rules of confidence based on Dempster-Shafer theory. Though they redefine argument types, their D-arg and FC-arg cases are similar to C argument and NSC/SC arguments in trust cases. Other aggregation approaches with similar issues are given in [1, 24, 25].

Alternatively, Noll et al. [15] in their Multi-Metrics approach, have claimed that quadratic functions reflect the aggregation better than linear approaches. Since security is as strong as its weakest link, instead of what is already good, Noll et al. focus on what is weak and what could go wrong, modeled with their concept of *criticality*. Then criticality is emphasized in the aggregation by evaluating the root mean square value of criticality values.

In our case, the arguments contribute individually to the overall goal. Based on the significance of the component, we assign appropriate weights to them. We have considered the weights in the range [0,100]. In this report, we compare the weighted mean approach and Multi-Metrics approach by calculating beliefs using both of the approaches. As an example to show how the arguments and beliefs are structured in the data encryption criterion, a GSN diagram for data encryption criteria is shown in Figure 4.

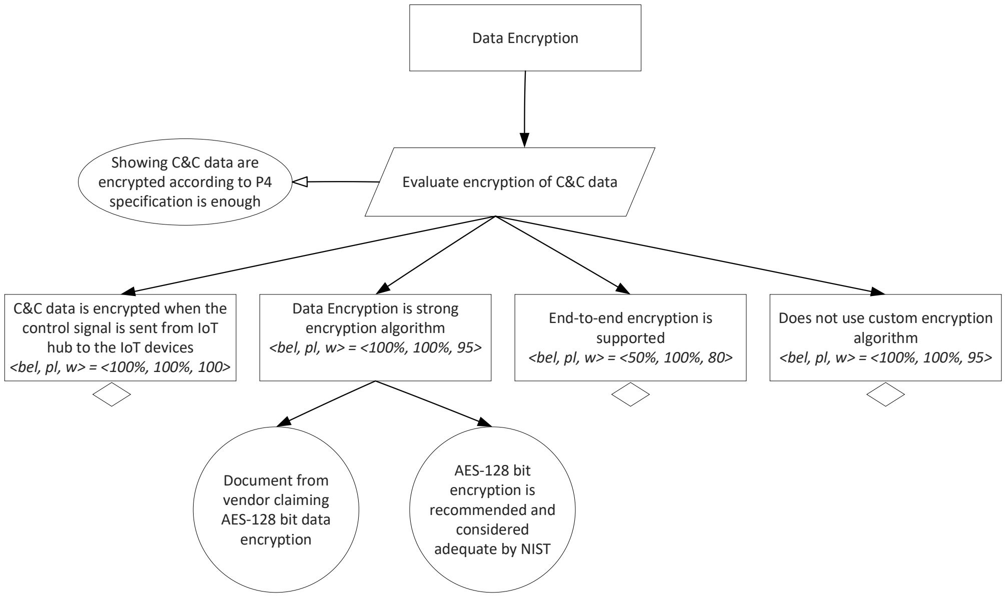

Data Encryption
Showing C&C data are encrypted according to P4 specification is enough
Evaluate encryption of C&C data
C&C data is encrypted when the control signal is sent from IoT hub to the IoT devices <bel, pl, w> = <100%, 100%, 100>
Data Encryption is strong encryption algorithm <bel, pl, w> = <100%, 100%, 95>
End-to-end encryption is supported <bel, pl, w> = <50%, 100%, 80>
Does not use custom encryption algorithm <bel, pl, w> = <100%, 100%, 95>
Document from vendor claiming AES-128 bit data encryption
AES-128 bit encryption is recommended and considered adequate by NIST

Figure 4: Evaluation of data encryption criterion for protection level 4 using Goal Structuring Notation

5.1.1 Weighted mean approach

Weights represent the significance. The weighted mean is calculated to obtain an average when the significance of values differ. The aggregated belief using the weighted mean for beliefs (b)

and weights (w) can be calculated using the formula:

$$AggregatedBelief(B) = \frac{\sum_i^n w_i b_i}{\sum_i^n w_i} \tag{6}$$

where w_i and b_i is the weight and belief of the i^{th} component.

5.1.2 Multi-Metrics approach

The Multi-Metrics (MM) approach of [15] uses Root Mean Square Weighted Data (RMSWD) to aggregate the criticality values. The formula to calculate the aggregated criticality is expressed as:

$$X = \sqrt{\sum_i \left(\frac{x_i^2 W_i}{\sum_i^n W_i}\right)} \tag{7}$$

where X is the aggregated criticality, x_i is the criticality of the i^{th} component, and W_i is calculated from the component weight w_i as:

$$W_i = \left(\frac{w_i}{100}\right)^2 \tag{8}$$

The original work focuses on criticality x_i defined as the complement of security, privacy, or dependability metrics [15]. In our context, we use the complement of belief value (*100 - belief*) to express criticality. Finally, the aggregated belief (Bel) is computed as a complement of X (i.e., $Bel = 100 - X$). Thus, using equation 7, Bel can be expressed as:

$$Bel = 100 - \sqrt{\sum_i \left(\frac{(100 - bel_i)^2 W_i}{\sum_i^n W_i}\right)} \tag{9}$$

where bel_i is the individual belief value of the component under consideration.

5.2 Aggregation of Security Classes

The classes obtained for each component should be aggregated to obtain an overall class to reflect the system security class. However, currently, the class is represented as a label (A-F). To allow aggregation between classes, we propose to have a scoring mechanism to represent the score of a security class. We assume that the consecutive classes are equidistant (i.e., the distance from class A to B is equal to that of B to C). Thus, initially, we mapped the higher score to a better class, i.e., the initial mapping between the score and the class is 1=F, 2=E, 3=D, 4=C, 5=B, and 6=A. However, we have used most of the parameters (beliefs, weights) in the range [0-100], and thus we normalize this score to the common range [0-100] (i.e., 0 maps to 1 and 100 maps to 6) for better perception. To transform the value from one scale to another, we use min-max normalization technique [7] as in Equation 10.

$$V_{norm} = \left(\frac{V - V_{min}}{V_{max} - V_{min}}(V_{newmax} - V_{newmin}) + V_{newmin}\right) \tag{10}$$

Where:

V_{norm} is the new normalized value and,

V is the given value and,

V_{max} is the maximum value in the existing scale and,

V_{min} is the minimum value in existing scale and,

V_{newmax} is the maximum value of the new scale and,

V_{newmin} is the minimum value of the new scale

To illustrate an example of a min-max normalization, let us suppose the obtained class is B, i.e., 5 according to the initial mapping. To represent it in a new scale between [0-100], we have $V = 5, V_{min} = 1, V_{max} = 6, V_{newmin} = 0, V_{newmax} = 100$. Using equation 10 the value 5 is transformed as:

$$V_{norm} = \left(\frac{5-1}{6-1}(100-0)+0\right) = 80$$

Table 2 shows the mapping of class to the normalized scores using equation 10. The table also shows the range of scores to which each of the classes is mapped. For example, if we obtain class B, then the score value would be 80 (see Table 2). Similarly, if the resulting class score after aggregation is 65, then it lies between the range [50,70], thus class C. We can now use the aggregation principle to obtain the overall class. Similarly to beliefs, the aggregation of security classes should be more sensitive to lower scores (lower class) than the higher scores. If one component has a lower class, despite other components with a higher class, the aggregated class should be closer to the lower class (supporting the weakest link principle of security).

Table 2: Security class mapping

Class label	Value	Normalized value	Range
A	6	100	90-100
B	5	80	70-90
C	4	60	50-70
D	3	40	30-50
E	2	20	10-30
F	1	0	0-10

5.3 Underlying principles for aggregation

Belief aggregation depends on how the arguments are presented. There are cases when there are multiple justifications independent of each other fulfilling the same claim, or sometimes each justification contributes towards the fulfillment of the claim to some extent. These should be handled differently. Here we describe the principles to guide the aggregation mechanism.

1. **Maximum belief**: If the justifications are overlapping and one includes another, the one with the highest belief should be selected. For example, to verify the claim "Data is encrypted", there are two claims with different beliefs. Document from the vendor describing that the data is encrypted (belief = 90%), Experimental verification that the data is encrypted (belief = 100%). In this case, we simply select the highest belief because the vendor document does not affect the overall belief as it has been proven experimentally.

$$AggregatedBelief = Maximum(b1, b2)$$

Here, b1 and b2 are beliefs on overlapping claims where justifying one of the claims includes another.

2. **Zero belief**: If the belief on any of the claims in protection level evaluation is zero, then the total belief is also zero. It is because the class is determined based on the previously specified requirements of security functionality. If one of the functionality has no belief at all, then the whole claim for that protection level fails, and it must be evaluated against the lower protection level. The same applies to the aggregation of protection criteria towards the protection level.

   $if(c.securityFunctionalities.Any(securityFunctionality.belief = 0))$
   $THEN\ c.belief = 0$

3. **Minimum belief**: Typically, it is assumed that the impacts and connectivity have full beliefs. If it has lower beliefs than 100%, the resulting belief should be the lowest one. For example, if the exposure obtained is E2 with a belief of 90% and the Impact is Major with 60%, the class obtained should have a belief of 60% instead of averaging it. It is because both of them are equally important and required for evaluation. Thus, the average of their belief has no meaning.

   $$AggregatedBelief = Minimum(b1, b2)$$

   Here, $b1$ and $b2$ are beliefs on impact and exposure claims.

6 Case study

To demonstrate the applicability of confidence in security classes, we have selected a use case about a home control unit in the SHEMS. Device control commands in SHEMS are used to manage the peak load by controlling the home appliances to reduce consumption. The scenarios for the use case are built upon our previous work [17], which used the two principal methods to control the IoT devices: centralized and edge control. The centralized control has higher connectivity and major impacts and thus results in class D, whereas in the edge control scenario, the connectivity towards control mechanism was reduced. Thus the impacts are also reduced, resulting in class A. We select the edge control scenario because it yielded a better security class (class A) than the centralized control (class D). By applying our enhanced methodology, we demonstrate how confidence in the solution is evaluated.

6.1 Protection level evaluation

For a device control command, data encryption, access control, and monitoring & analysis were considered the relevant criteria. We first analyzed the security mechanisms available for each security criteria of Command and Control (C&C) data and mapped it to the protection level. The security mechanism of the C&C component mapped to the protection level is summarized in Table 3. The column 'P' shows the existing security functionalities in the C&C component. This column maps to the P4 column, therefore, the protection level is considered to be P4. Next, we discuss confidence parameters for each security criteria and their security mechanisms.

6.1.1 Data encryption

To evaluate the data encryption criterion in the edge control scenario, we consider the C&C data in the home network. The following sub-claims were considered to satisfy the P4 level requirements:

Table 3: Protection level requirements for C&C data

Protection criteria	Security functionality	P5	P4	P3	P2	Eval.
Data encryption	Encryption of C&C data between IoT hub and devices	x	x	x	x	✓
	Encryption of C&C data between IoT hub and backend system	x	x	x		✓
	Strong encryption mechanism	x	x	x		✓
	Credentials should not be exposed in the network	x	x	x		✓
	End-to-end encryption	x	x			✓
	Should not use custom encryption algorithms	x	x			✓
Access control	Disable remote access functionality	x				
	Default and weak passwords should not be used	x	x	x		✓
	Enable multi-factor authentication	x	x			N/A
Monitoring and analysis	Monitoring system components	x	x			✓
	Analysis of monitored data	x	x			✓
	Act on analyzed data	x				

- **C&C data is encrypted when the control signal is sent from the IoT hub to the IoT devices**: The belief on this claim is 100% justified by the vendor's document and lab test.
- **Data encryption uses a strong encryption algorithm**: It has been verified that data is encrypted with AES 128-bit encryption, which is considered secure for the home network. Thus, the belief in this sub-claim is also set to 100%
- **End-to-end encryption is supported**: Communication uses Zigbee, which supports end-to-end encryption. However, we did not find any claims from the vendor about it. Thus, this claim is partially trusted (50%) but would gain the plausibility of 100% if verified experimentally or claimed by the vendor.
- **Does not use a custom encryption algorithm**: This claim has 100% belief because the communication uses the Zigbee standard, which supports a well known AES-128 bit encryption.

6.1.2 Access control

In our case, the C&C command is sent from the IoT hub to the devices in the home network, based on the preset threshold. We consider the following two sub-claims to fulfill P4:

- **Weak and default credentials should not be used**: The hub and the devices are authenticated using unique pre-shared keys, and thus unauthorized access of C&C data is not allowed. The C&C data has restrictions to be accessed and triggered only by the gateway. Thus, we consider access control as adequately secured and assign the belief of 100%.
- **Enable multi-factor authentication**: Not relevant

6.1.3 Monitoring and analysis

In protection level P4, monitoring and analysis criteria require two security functionalities: monitoring system components and analysis of monitored data. Thus, the claim for this criterion "Monitoring and analysis are adequate" can be supported by the following two sub-claims:

- **C&C data is adequately monitored**: The advanced monitoring capabilities include the collection of security data from system components, analyze them, and take necessary actions when required. In our case, it supports basic monitoring functionalities where log information such as device status, control signals sent and executed are collected. Thus, the belief assigned to the first sub-claim "Data is adequately monitored" of C&C related data is 98%.
- **C&C data is adequately analyzed**: The gateway performs regular availability checks on its devices and notifies any device disconnection. However, it does not perform any extensive automated analysis of abnormal behavior of control signals. It is possible to manually analyze the monitored data from the log occasionally. Yet, it does not perform extensive security analysis. Thus, the sub-claim "data is adequately analyzed" has a lower belief set to 80%.

Table 4 summarizes the beliefs, plausibilities, and weights assigned to the parameters for protection level evaluation of the selected criteria.

Table 4: Summary of belief and plausibility.

Protection criteria	**Security functionality**	**bel, pl, w**
Data encryption (w=100)	C&C data is encrypted when the control signal is sent from IoT hub to the IoT devices	<100, 100, 100>
	Data encryption uses a strong encryption algorithm	<100, 100, 95>
	End-to-end encryption is supported	<50, 100, 80>
	Does not use custom encryption algorithms	<100, 100, 95>
Access control (w=95)	Weak and default credentials are not be allowed	<100,100,100>
Monitoring and analysis (w=80)	Monitoring system components	<98, 100, 100>
	Analysis of monitored data	<100,100, 95>

6.2 Aggregation of beliefs

As mentioned in Section 5.1, we consider the weighted mean approach and MM approach for belief aggregation. In our case, there was no disbelief on any of our premises, and thus we consider 100% plausibility (See equation 4).

6.2.1 Aggregation using the weighted mean approach

Using this approach, we calculated the aggregated belief (Equation 6) for data encryption, access control, and monitoring & analysis as 89%, 100%, and 89%, respectively. Further aggregation towards protection level evaluation gave us 93% belief and 7% uncertainty. Evaluating both the connectivity and protection level is equally important, and in the absence of one of the parameters, exposure cannot be evaluated. Thus, if the belief in connectivity evaluation is fully trusted, the resulting confidence of exposure is determined by the confidence in the protection level. The same applies to impact and exposures. Thus, we assign the overall confidence to the class A evaluation as 93% belief and 7% uncertainty (i.e., $< A, 93, 7 >$)

6.2.2 Aggregation using the MM approach

Using this approach (see Equation 9), the aggregated belief value for data encryption, access control, and monitoring & analysis criteria obtained were 78%, 100%, and 86%. Similarly, the aggregated belief for P4 using the MM approach was 84%. Since plausibility was considered 100% all the time, it does not change after aggregation. The class obtained using the MM approach gave us class A with 84% belief and 16% uncertainty, i.e., $< A, 84, 16 >$, which is more realistic than the weighted mean approach which obtained $< A, 93, 7 >$ despite of the lower belief on one of the sub-claim. The MM approach retains the lower belief values in the result. Table 5 summarizes the results obtained from the two approaches.

Table 5: Comparison of weighted mean and RMSWD approach for belief aggregation.

Weighted mean approach		**Protection criteria**	**Multi-Metrics approach**	
Aggregated to protection level	**Aggregated to criterion level**		**Aggregated to criterion level**	**Aggregated to protection level**

Bel = 93%
Pl = 100%
U = 7%
	Bel = 89%
	Pl = 100%
	U = 11%
		Data encryption (w=100)	Bel = 78%
			Pl = 100%
			U = 22%%
				Bel = 84%
				Pl = 100%
				U = 16%

	Bel = 100%
	Pl = 100%
	U = 0%
		Access control (w=95)	Bel = 100%
			Pl = 100%
			U = 0%	

	Bel = 89%
	Pl = 100%
	U = 11%
		Monitoring & analysis (w=80)	Bel = 86%
			Pl = 100%
			U = 14%	

7 Discussion

We have compared two approaches to aggregate beliefs: the weighted mean and the MM approach. The aggregation in the weighted mean approach is not sensitive to lower values. For instance, when the data encryption criterion was evaluated, the beliefs for security functionalities were 100, 100, 50, 100, and their respective weights were 100, 95, 80, and 95. Here, one of the beliefs with weight 80 has a low belief value of 50. However, the aggregated belief is calculated as 89%. This value is not very realistic because in security, if one of the claims has lower belief, it may have a profound effect on the overall security, and thus lower values should be well reflected in the aggregation of beliefs in security. For the same criterion (i.e., Data encryption), when applying the MM approach to aggregate the belief, we obtained an aggregated value of 78%, which is somewhat more realistic than 89%. Thus, we prefer the MM approach to aggregation of beliefs instead of weighted means. The overall result of belief using the MM approach showed that the overall belief for the C&C data being Class A was 84%, and the plausibility obtained was 100%. It means we could have obtained a total of 100% belief. However, due to a lack of evidence and knowledge, there was 16% uncertainty. This uncertainty can be reduced by providing missing evidence. For example, the belief in the existence of end-to-end encryption is 50%. It can be increased by gathering more evidence, i.e., experimental verification of end-to-end encryption, which could raise the overall belief to 93%.

It is also important to notice that plausibility is 100%, which means that in the given setup, for the C&C data there is no disbelief in the claims. Higher disbelief may suggest that the parameter causing disbelief must be changed because it is not trusted to a large extent. As an example, a claim is made for adequate data encryption for Wi-Fi communication. If it uses WEP standard, then the disbelief in the claim is high because, even though WEP provides encryption, it is known to be weak. Hence, to reduce the disbelief, the WEP must be replaced with a more secure standard such as WPA2.

The degree of belief of the result represents the trust in the claim made for a given class. If the belief is too low, and the uncertainty is high, then the assessment requires more work or the experts may have less knowledge about the security built around the system. However, if the belief is low and the disbelief is high, the claims made in the assessment are not trustworthy, and appropriate measures should be taken to improve the confidence. Similarly, an acceptable but not too high value of beliefs may say that the claims are trustworthy but not entirely acceptable. In terms of security class, it may mean a different level of trust in the assessment. For instance, a claim of class A with belief 60% and plausibility 95%, may mean a less trusted class A (e.g., A^{--}). However, in the same condition, belief with 90% may represent a highly trusted class A (e.g., A^{++})).

8 Tool support for security classification methodology

To demonstrate the applicability of security classification methodology, we needed a tool supporting argument structuring and confidence assessment. Tools such as ASCE [14], Certware [3], or D-case editor [13] support structuring of arguments. However, those tools do not support confidence assessment. Some of the confidence aggregation frameworks perform confidence aggregation [1, 22]. However, the way the confidence is aggregated may not be usable for security cases. NOR-STA tool [5] seams to provide most of the features that we need, including argument structuring and confidence assessment. We describe the tool briefly.

8.1 NOR-STA

NOR-STA is a tool implementation of trust cases providing a visual assessment of arguments. This tool's argument structure is based on Toulmin's argument model and is used to specify the claims and arguments as a tree structure. NOR-STA can also generate GSN diagram from the arguments. The confidence assessment is based on the Dempster-Shafer theory of belief. However, they introduce a linguistic decision and confidence scale to present the opinion towards presented arguments. This scale is then converted to Dempster-Shafer belief and plausibility, which are then used to compute the aggregated belief and plausibility. The aggregated beliefs and plausibility are then converted back to confidence and decision to show the result.

8.1.1 Scale of assessment

Trust cases are based on the Dempster-Shafer theory of belief and plausibility. However, assigning belief and plausibility to any argument is challenging and less intuitive. Thus, trust cases provide decision and confidence scales, which are relatively more intuitive. The decision scale shows the degree of acceptance or rejection of the assessed element. It has four decision values, namely *acceptable*, *tolerable*, *opposable*, and *rejectable*.

Similarly, the confidence scale shows confidence in the decision and has six values: *for sure*, *with very high confidence*, *with high confidence*, *with low confidence*, *with very low confidence*, and *lack of confidence*. The combination of decision and confidence scale shows the experts' attitude towards an argument. If the confidence is lowest (i.e., *no confidence*), it represents total uncertainty in the decision because whatever decision is made, it ceases to provide confidence in that decision. The authors map these decision and confidence scale to the belief (b), disbelief (d) and uncertainty (u) parameters as follows, where $b, d, u \in [0, 1]$:

$$b + d + u = 1.$$

Figure 5 represents the assessment in decision and confidence scale being transformed into belief, disbelief, and uncertainty scale. Each assessed element in the NOR-STA tree can be

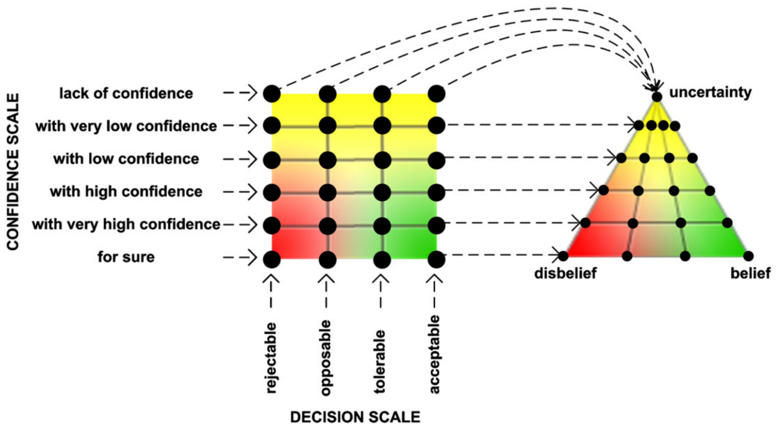


Figure 5: The assessment triangle [5]

aggregated towards the parent element based on the assigned argument aggregation rule.

This assessment can be transformed to Dempster-Shafer belief and plausibility function as the following equations:

$$\begin{aligned} Bel(s) &= Conf(s).Dec(s) \\ Pl(s) &= 1 - Conf(s).(1 - Dec(s)) \end{aligned} \tag{11}$$

where:

- s is a statement
- $Bel(s) \epsilon [0, 1]$ is the belief function that supports s
- $Pl(s)\epsilon[0, 1]$ is the plausibility function (upper bound on the belief in s that can be gained by adding new evidence).

8.1.2 Arguments and their aggregation in trust cases

In NOR-STA, each sub-claims can be further broken down into sub-argument models forming a tree structure. The leaves of the NOR-STA argumentation tree are the facts and evidence supporting its parent claim. NOR-STA provides the mechanism to aggregate arguments from the leaf level to the root. This tool considers three types of arguments and their aggregation rules, as described in [5]. Below are the aggregation rules being applied for each type of arguments [5] where:

- c is a claim (conclusion),
- w is the warrant of c,
- n is the number of premises/argument strategies of c,
- a_i (where $i\epsilon \{1, 2, ..., n\}$) is the ith premise,
- k_i (where $i\epsilon \{1, ..., n\}$ is the weight assigned to the ith premise

NSC-argument(Necessary and Sufficient Condition list argument):
For NSC-arguments, all the premises must be accepted for the conclusion to be accepted. Failing one premise rejects the whole conclusion. NSC-arguments are aggregated as:

$$Bel(c) = Bel(w) \cdot Bel(a_1) \cdot Bel(a_2) \cdot ... \cdot Bel(a_n) \tag{12}$$

$$Pl(c) = 1 - Bel(w) \cdot [1 - Pl(a_1) \cdot Pl(a_2) \cdot ... \cdot Pl(a_n)] \tag{13}$$

SC-argument (Sufficient Condition list argument):
For SC-arguments, if a premise fails, nothing can be inferred about the conclusion. The acceptance case of SC-arguments is the same as NSC-arguments. Thus, SC-arguments can be aggregated as:

$$Bel(c) = Bel(w) \cdot Bel(a_1) \cdot Bel(a_2) \cdot ... \cdot Bel(a_n) \tag{14}$$

$$Pl(c) = 1 \tag{15}$$

C-argument (Complimentary arguments):
In C-argument, the falsification of a premise does not reject the whole conclusion but reduces the confidence in the decision. This uses weight-based aggregation, and the aggregated assessment is a type of a weighted mean value of all premises. C-arguments are be aggregated as:

$$Bel(c) = Bel(w) \cdot \frac{k_1 Bel(a_1) + k_2 Bel(a_2) + ... + k_n Bel(a_n)}{k_1 + k_2 + ... + k_n} \tag{16}$$

$$Pl(c) = 1 - Bel(w) \cdot \left(1 - \frac{k_1 Pl(a_1) + k_2 Pl(a_2) + ... + k_n Pl(a_n)}{k_1 + k_2 + ... + k_n}\right) \tag{17}$$

A-argument (Alternative arguments):
In this type of argument, several independent arguments support the conclusion. The aggregation rule for A-Arguments is expressed as:

$$\begin{aligned} Bel(c) = {} & Bel(ar_1).Bel(ar_2) + Bel(ar_1)[Pl(ar_2) - Bel(ar_2)] + \\ & Bel(ar_2)[Pl(ar_1) - Bel(ar_1)] \end{aligned} \tag{18}$$

$$\begin{aligned} Pl(c) = 1 - \{ & [1 - Pl(ar_1)].[1 - Pl(ar_2)] + [1 - Pl(ar_1)].[Pl(ar_2) \\ & - Bel(ar_2)] + [1 - Pl(ar_2)].[Pl(ar_1) - Bel(ar_1)]\} \end{aligned} \tag{19}$$

NOR-STA considers a simple decision and confidence scale for assessment. Since these scales are not evenly distributed, calculations cannot be performed using the uniformly distributed scale. Thus, the authors experimentally calibrated the values using scaling function s define as:
$s : [0, 1] \times [0, 1] \rightarrow [0, 1] \times [0, 1]$ where, for each aggregation rule, the decision and confidence value of range [0,1] maps to a new value within the same range [0,1].

The trust case uses a combination of linear normalization equations to compute the scaled value. After aggregated values are calculated, the inverse scaling function should be used to convert these scaled values back to the intuitive values. The paper claims that the scaling function was constructed based on the experimental evaluation, however, it does not provide values for the constants of scaling functions, and thus the results obtained from the tool could not be replicated when calculated manually.

There are four different aggregation rules applied to the tool to propagate beliefs. The issues with C-Arguments, SC-Arguments, and NSC-Arguments have been discussed in section 5.1

Thus, though NOR-STA is good at structuring the arguments, the aggregation principles do not fit our context. Therefore, we did not use this tool to apply the security classification method.

9 Conclusion and future work

In this work, we have enhanced the security classification method proposed in [19] by introducing confidence parameters (i.e., belief and uncertainty) to the assessment of a security class. Specifying confidence in the assessment improves the trustworthiness and quality of the assessment. We follow the GSN structured approach to organize the security arguments, making it easier to assign the confidence parameters. We then exemplified how to calculate confidence parameters for a use case involving an edge command & control mechanism for SHEMS. We also discuss how to assign overall confidence in the assessment results. By comparing two types of aggregation methods, we observed that the weighted average methods are less suitable than the method based on RMSWD, which is used to aggregate the criticality in the MM method of [15]. We also discussed different principles guiding when to consider minimum, maximum, and zero belief during the aggregation.

Further work is required for building these argumentation and aggregation methods into a tool, following works on tools like NOR-STA [5]. This is challenging to integrate with the security classification methodology that comes with predefined mechanisms and lookup tables.

References

[1] Anaheed Ayoub, Jian Chang, Oleg Sokolsky, and Insup Lee. Assessing the overall sufficiency of safety arguments. 2013.

[2] Gianmarco Baldini, Antonio Skarmeta, Elizabeta Fourneret, Ricardo Neisse, Bruno Legeard, and Franck Le Gall. Security certification and labelling in internet of things. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 627–632. IEEE, 2016.

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

[4] Peter Bishop, Robin Bloomfield, and Sofia Guerra. The future of goal-based assurance cases. In *Proc. Workshop on Assurance Cases*, pages 390–395. Citeseer, 2004.

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

[6] AP Dempster et al. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 38(2):325–339, 1967.

[7] Salvador García, Julián Luengo, and Francisco Herrera. *Data Preparation Basic Models*, pages 39–57. Springer International Publishing, 2015.

[8] John Goodenough, Howard Lipson, and Chuck Weinstock. Arguing security-creating security assurance cases. *rapport en ligne (initiative build security-in du US CERT), Université Carnegie Mellon*, 2007.

[9] Jean Gordon and Edward H Shortliffe. The dempster-shafer theory of evidence. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, 3:832–838, 1984.

[10] Trudy Govier. *A practical study of argument.* Cengage Learning, 2013.

[11] Tim Kelly. Reviewing assurance arguments-a step-by-step approach. In *Workshop on assurance cases for security-the metrics challenge, dependable systems and networks (DSN)*, 2007.

[12] Mike Maksimov, Nick LS Fung, Sahar Kokaly, and Marsha Chechik. Two decades of assurance case tools: a survey. In *International Conference on Computer Safety, Reliability, and Security*, pages 49–59. Springer, 2018.

[13] Yutaka Matsuno. D-case editor: A typed assurance case editor. *University of Tokyo*, 2011.

[14] Kateryna Netkachova, Oleksandr Netkachov, and Robin Bloomfield. Tool support for assurance case building blocks. In *International Conference on Computer Safety, Reliability, and Security*, pages 62–71. Springer, 2014.

[15] Josef Noll, Inaki Garitano, Seraj Fayyad, Habtamu Abie, et al. Measurable security, privacy and dependability in smart grids. *Journal of Cyber Security and Mobility*, 3(4):371–398, 2014.

[16] Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976.

[17] Manish Shrestha, Christian Johansen, and Josef Noll. Criteria for security classification of smart home energy management systems. In *International Conference on Smart Information & Communication Technologies.* Springer, 2019.

[18] Manish Shrestha, Christian Johansen, and Josef Noll. Building confidence using beliefs and arguments in security class evaluations for iot. In *The Fifth International Conference on Fog and Mobile Edge Computing (FMEC 2020).* IEEE, 2020.

[19] 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.

[20] John Spriggs. *GSN-The Goal Structuring Notation: A Structured Approach to Presenting Arguments.* Springer Science & Business Media, 2012.

[21] Stephen E Toulmin. *The uses of argument.* Cambridge university press, 2003.

[22] Rui Wang, Jérémie Guiochet, and Gilles Motet. Confidence assessment framework for safety arguments. In *International Conference on Computer Safety, Reliability, and Security*, pages 55–68. Springer, 2017.

[23] Rui Wang, Jérémie Guiochet, Gilles Motet, and Walter Schön. Safety case confidence propagation based on dempster–shafer theory. *International Journal of Approximate Reasoning*, 107:46–64, 2019.

[24] Goitom Kahsay Weldehawaryat and Basel Katt. Towards a quantitative approach for security assurance metrics. In *The Twelfth International Conference on Emerging Security Information, Systems and Technologies; SECURWARE 2018 September 16, 2018 to*

September 20, 2018-Venice, Italy. International Academy, Research and Industry Association (IARIA), 2018.

[25] Shuichiro Yamamoto. Assuring security through attribute gsn. In *2015 5th International Conference on IT Convergence and Security (ICITCS)*, pages 1–5. IEEE, 2015.

Paper IV

Security Classification for DevSecOps: Five Principles and an Exemplification

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

Submitted to ACM Transactions on Privacy and Security (TOPS)

A technical report (starting at page 157) supporting this publication was published in September 2020, with the research report number 495, ISBN 978-82-7368-460-8, and ISSN 0806-3036.

The report can be found at:

http://urn.nb.no/URN:NBN:no-84145

Security Classification for DevSecOps: Five Principles and an Exemplification

MANISH SHRESTHA, eSmart Systems AS
CHRISTIAN JOHANSEN, Department of Technology Systems, University of Oslo
MAUNYA DOROUDI MOGHADAM, Department of Technology Systems, University of Oslo
JOHANNA JOHANSEN, Department of Informatics, University of Oslo
JOSEF NOLL, Department of Technology Systems, University of Oslo

DevSecOps is the extension of DevOps with security training and tools included throughout all the phases 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 such as risk analysis or certification. As such, security classifications are not compatible with DevSecOps, which primarily goes away from the traditional style of penetration testing, done only when the software product is in the final stages or already deployed.

In this work, we first identify five principles for a security classification to be *DevOps-ready*, two of which are the focus for the rest of the paper, namely to be tool-based and easy to use for non-security experts, such as ordinary developers or system architects. We then exemplify how one can make a security classification methodology DevOps-ready. We do this through an interaction design process, where we create and evaluate the usability of a tool implementing the chosen methodology. Such work seems to be new within the usable security community, and even more so in the software development (DevOps) community. Therefore, we extract from our process a general 'recipe' with three points that others can follow when making their own security methodologies DevOps-ready. The tool that we build is in itself a contribution of this process, as it can be independently used, extended, and/or integrated by developer teams into their DevSecOps tool-chains. Our tool is perceived as most useful in the design phase, but also during the testing phase where the security class would be one of the metrics used to evaluate the quality of their software.

CCS Concepts: • **Human-centered computing → User centered design**; • **Security and privacy → Usability in security and privacy**; *Software security engineering*; • **Software and its engineering → Agile software development**; *Programming teams.*

Additional Key Words and Phrases: DevOps, usable security, usability testing, security classification tool, Internet of Things

ACM Reference Format:
Manish Shrestha, Christian Johansen, Maunya Doroudi Moghadam, Johanna Johansen, and Josef Noll. 2021. Security Classification for DevSecOps: Five Principles and an Exemplification. In *ACM CHI Conference on Human Factors in Computing Systems, May 8-13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 22 pages. https://doi.org/??

1 INTRODUCTION

According to International Data Corporation, the predicted number of Internet of Things (IoT) devices for 2025 is 41.6 billion, generating ca. 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 driven by functionality and low costs, demanded by the high competition in this new market, as argued, e.g., by [22, 25, 34].

IoT software, like most modern software, is developed in an agile style (see e.g., the Scrum[2] method), where popular now is the DevOps culture [8]. DevSecOps[3] adds security tools and awareness at all phases of the software development life-cycle [17]. However, security tools [23, Part VI] need to have low learning and usability thresholds before they can be effectively included in the DevOps tool-chain [12].

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, risk analysis or security classification cannot keep up with the fast changing threat landscape in IoT systems [27]. 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[4] or BSI Kitemark[5] are mostly based on penetration testing and risk analysis, besides documentation. Risk assessment methods (s.a., CORAS [14], EBIOS[6], FAIR [20], and OCTAVE [1]) require significant amounts of time and resources to conduct. These approaches follow a waterfall model where the assessments are far less frequent than the releases, and thus cannot fit the agile style of system development [13].

As such, the software industry (and especially the IoT one) lacks motivation and guidelines for building security by design. We think that DevSecOps is one positive drive 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 have only recently started to appear (see e.g., IoTSF[7], GSMA[8], IoT Working Group of CSA[9], or the Industrial Internet Consortium[10]).

Contributions. Based on literature and our experience with security classifications and DevOps practices, we identify (for the first time) five principles 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 Sec. 2.2). We then choose an existing security classification methodology that already satisfies (4) and (5) from [31], and focus here on making it satisfy the principles (2) and (3). Since the first principle is dependent on (2), we discuss it as future work.

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

[1]https://www.idc.com/getdoc.jsp?containerId=prUS45213219
[2]https://ScrumGuides.org
[3]https://www.devsecops.org
[4]https://ims.ul.com/IoT-security-rating
[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.iotsecurityfoundation.org/wp-content/uploads/2018/12/IoTSF-IoT-Security-Compliance-Framework-Release-2.0-December-2018.pdf
[8]https://www.gsma.com/iot/iot-security-assessment/
[9]https://downloads.cloudsecurityalliance.org/assets/research/internet-of-things/future-proofing-the-connected-world.pdf
[10]https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB.pdf

managers, or personnel from software operations. Our users, described more thoroughly in Sec. 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 our development; as well as (ii) SMEs from a Polish cluster, and (iii) several developers recruited from software developing companies, both groups involved only for evaluating our high-fidelity web-based prototype. To evaluate our prototypes and to extract information from our users, we organized workshops during which usability studies were run, involving methods such as interviews, observations, co-design, and active intervention, as well as standard questionnaires and recordings of user actions.

We do our work in five stages, developing three prototypes along the way; this is what we describe in Sec. 4 (the manual stages) and Sec. 5 (the tool prototypes). In the end, we extract from this process as a "recipe" to make it easy for others to transform security classifications (or similar methods) into DevOps-ready tools, by following and maybe adapting our stages and instruments. We strive to make these stages intuitive and natural, following interaction design principles, but applied to our particular task of 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 Sec. 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) [30] and AMI (Advanced Meeting Infrastructure) [31]. Then one needs to transform the methodology into a process (steps to follow) focused on the non-expert target users (see Sec. 4.3). The process then should be implemented into a low-fidelity prototype, e.g., in our case using spreadsheets (see Sec. 5.1), to test the automation and procedure nature of the method. From the evaluation of the first implementation, one draws more concrete requirements for the high-fidelity version (see Sec. 5.2). In the end we evaluate (see Sec. 5.3) our final candidate for integration into a DevOps tool-chain.

Our current and future work is to help the software company eSmart Systems AS (which provides cloud-based solutions for smart grid monitoring of AMI) take up into their development process the tool that we present in this paper. 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 their specific development process and to enable dinamicity.

2 SECURITY CLASSIFICATION FOR DEVSECOPS

2.1 DevSecOps and Usability of Security

Agile methods [6] have been a popular style of software development for quite a while, adopting from the spiral model [4] the cyclic development, revisiting the same phase multiple times, e.g., new or changed requirements may be dictated by the client or the market. Agile promotes the inclusion of users, e.g., their manifesto[11] encourages a software development culture that values: (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.

DevOps is a more recent agile style that differentiates itself through being open to and encouraging the use of tools at all phases, including the operations phase (thus the 'Ops' in the name). Operations have become more important lately, both because of the proliferation of the cloud, making the infrastructure cheaper to deploy and run the software, and because of automation and tools becoming available for more tasks in all the development phases. DevSecOps brings into the DevOps the security, following the same philosophy, i.e., security awareness (or best practices) and security tools/processes at all phases. In particular, the penetration testing that depends on a high level of security expertise

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

(usually coming from outside the development team) is mostly replaced by security tools such as code scanners, loggers, or API security testing, and phase relevant security education for all team members.

We consider DevSecOps as an arena that, more than ever, promotes the industrial adoption of usable security tools [? ?]. On one hand, since DevSecOps is so tool intensive it lowers the usability threshold to allow more tools to be incorporated into the development tool-chain. On the other hand, since DevSecOps is so receptive to new tools, it offers researchers a motivation to put more effort into making their security tools easier to use, in the hope of being adopted by the industry.

2.2 Principles for DevOps-ready Security Classifications

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

If a reader not acquainted with security classifications may have difficulties following some of the, rather succinct, arguments behind the five principles, we trust that after going through the details of Sec. 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; 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: *Impact* and *Likelihood* (that 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. (See also Fig. 1 on page 7.)

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[12] applications (e.g., nowadays web browsers[13]) 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., no consent). Therefore, any security classification needs to be dynamic so that for each update, quick and cheap re-evaluations can be done – similar to how software testing is being done – to cope with the short development life-cycles of DevOps.
(2) **Tool-based.** The method must have a tool support, and not only with a GUI but also with an API available, so that is can be integrated within the overall DevSecOps tool-chain (e.g., [16]). Tools that are built with a UI (e.g., web-based apps) are also built with an API (e.g., RESTful) to which the UI connects, thus, an API is most often a byproduct of the tool development.

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

(3) **Easy to use for non-security experts.** 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 (often already deployed) system, and into a new style where every member of the DevOps team needs to have security competence relevant for their phase of development. Thus, a security classification method for DevSecOps needs to be usable by non-security experts.

(4) **Impact statically and manually evaluated.** Security classifications (the same as risk analysis methods) involve evaluating the impacts of security breaches. However, when using the security classification inside one company for developing one product, the impact evaluation is nearly static because the planned product and its functionalities and intended use, do not change radically during the lifetime of the product. Changes are usually very controlled, and those that are relevant for the evaluation of impact are even less frequent. As such, the security methodology is enough to evaluate impacts once, in the beginning (maybe using even security experts), and input this evaluation manually into the tool. Therefore, we assume that impacts are of little concern for a DevOps-ready tool, and one need not spend effort on automating that.

(5) **Fine-grained security functionality.** Outside impact, security classifications are usually attack-centric, focusing on the capabilities of the attackers. For IoT and for a DevOps style of development, one would 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, which are under the full control of the DevOps team. Focusing on functionalities makes it easy to automatically evaluate the system within a DevOps testing cycle, 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 will work with is already developed to meet requirements 4 and 5. Thus we do not evaluate these here. Moreover, the dynamicity requirement can be achieved and evaluated only after a tool is built (see Sec. 6). Therefore, in this paper, we focus on the two principles 2 and 3.

3 PARTICIPANTS

The research in the paper has a user-centered approach, where the users and their goals are the driving force behind the development of a Security Classification Tool (SCT). Usability testing [11] helps us discover problems with the chosen SCM and to develop an easy to use SCT for non-security experts.

Our target group is *non-security experts*, motivated by Principle 3. More precisely, we are interested in people that have technology expertise, as well as people, such as system designers and developers, who are not security engineers but who may have basic security training (since their routine tasks need this) 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; these people would know about use-cases, features, or economy and impacts, related to the technology system, but not necessarily technical details.

The participants involved in testing our prototypes are:

SCOTT project. The most inputs and interactions were done with the participants from one large project called Secure Connected Trustable Things[14] (SCOTT) with "57 partners from industry and academia from 12 countries working on 15 pilots involving 48 technological building blocks". The main companies that we interacted with

[14]https://scottproject.eu

were: Philips Research[15] (NL), Vemco[16] (PL), AVL[17] (AT), ISEP[18] (PT), VTT[19] (FI) and Tellu IoT[20] (NO), as well as academics from Gdansk University of Technology[21] (PL) and KTH (SE).

Students. These were attending one course on IoT security. There were relatively few student participants, but their inputs were valuable and representative for their target group (i.e.,the novice users).

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

Software experts. We also reached out to four individuals from industry who had long software development experience:

- Participant 1: CEO of a startup company with more than 25 years of experience in the software industry, especially on 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.

In particular 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. Rather to the contrary, the 'Students' are technical people with little knowledge of security and fresh in the development field too. The 'SME cluster' was chosen so we can have teams that are more diverse in expertise, from business experts to developers (detailed in Sec. 5.3.1).

In our studies we were interested in testing with both individual users working alone (i.e., the 'Students' and 'Software experts'), but also with teams where the members collaborate in using the SC tool (i.e., the SCOTT and the SME cluster participants). Since our aim is to provide a SC tool for the DevSecOps team, both team work and individuals are important, as well as diversity of background, e.g., spanning the design, development, as well as the operations phases of DevOps. Our hackathon from Section 5.3.1 is especially focused on diversity, whereas involving the individual 'Software experts' in Section 5.3.2 is meant to reach various types of DevOps work.

The users have been consulted throughout the development, and we explain in the rest of the paper how and for which of our studies we interacted with the different users from above.

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 [31] as an extension of the standard for "Security Classification of Complex Systems" developed by the French national agency

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

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

ANSSI. Besides, the methodology of [31] incorporates security concepts from (and conforms with) several other relevant standards, among others, ISO/IEC, ETSI, OWASP, ENISA. This method has been detailed and extended towards IoT systems in [29]. We give here a very short review of this specific SCM, since more details will appear in Sec. 4.3.

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 Fig. 1. SCM considers five levels of Connectivity [31, Sec.3.1] adopted from ANSSI.

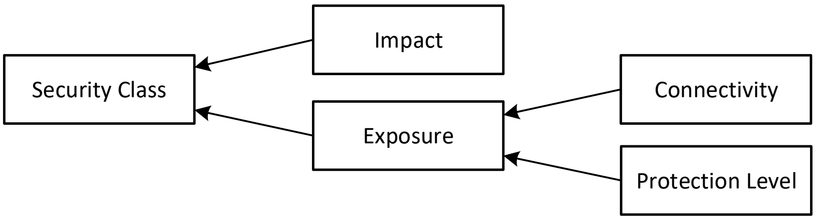


Fig. 1. Components of the evaluation of a security class.

The protection mechanisms are evaluated based on a list of security criteria [30, 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 [30, Sec.3.7]) – namely Insignificant, Minor, Moderate, Major and Catastrophic. The impact level is determined usually by security experts (as mentioned by Principle 4 in Sec. 2.2).

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.

4.2 Evaluating the SC Methodology with Users

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

The goal of the first stage is to take the methodology as described in the research papers [29–31] and evaluate whether it follows the Principle 3, i.e., that the SCM is easy-to-use for non-experts in security.

The Participants in this evaluation stage were from two of our user groups, namely the Students and partners from the SCOTT project who 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.

Performing the test: Our research team, which included security experts, first read relevant papers and understood from [31] the SCM. We then prepared one presentation for the two groups of users. (A) To the SCOTT partners, we presented and explained the SCM through a one hour workshop. The feedback was collected through structured conversations during a session after the presentation. (B) To the students, we presented the SCM in one of the lectures and gave as a homework the research papers, which they were supposed to apply to their IoT system exercise and report back to the lecturer (one of our research team members).

The first results were that none of the participants could understand the SCM, let alone how to apply it to their use cases. However, they did express interest in the concept of security classes. We did not obtain more concrete suggestions, mainly because the participants could not understand enough about SCM to give us meaningful comments.

Our team took then a second attempt at simplifying the presentation, and more importantly, we now presented how the SCM would be applied, focusing on exemplifying the work published in [30]. 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 read various SCOTT project documents where their respective IoT systems were being described. We then tried in our presentation to make, rather superficial, correlations between the application of the SCM from [30] and the participants' respective pilot systems. This second workshop with SCOTT did not manage to clarify enough as to allow the participants to apply the SCM. However, we did get more feedback during the structured conversations session. The topics included details of the SCM, like the calculation of impact and the evaluation of connectivity.

The final result can be summarized, based on one of the participants observations, endorsed rather unanimously, as

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

Explanations and Recommendations: When reflecting on this observation, we could correlate it with how certification bodies use certification processes to do their work. An elementary definition of 'process' implies a sequence of steps to be followed to arrive at a desired outcome. Having a predefined process for users to follow resonates with the external cognition approach [28]. Externalizing to reduce cognitive load means, in our case, producing a sequence of steps that a non-security expert could follow in order to evaluate the security class that a system belongs to. Following the cognitive tracing technique, we decided to create a step-by-step process, meant to organize and externalize the requirements of the methodology and guide the users through the actions needed to perform a classification.

4.3 SC methodology as a ten-step process

We have structured 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, e.g.: system architecture, functionalities, security requirements, use cases, and context of use. This step helps the user to understand and prepare the system under evaluation.
(2) **Define the components of the system.** The necessary components of the system are defined, e.g., for a smart home one can have: IoT hub, smart devices, sensors, control data, etc.

(3) **Describe the features of the components.** The interactions between the system components are described. By now the user should have a reference architecture of the system and have identified a use case.
(4) **Define the impact level.** For each component, the worst impact of security breaches is defined. The impact levels are defined by the SCM research papers (following ANSSI) and is similar to the evaluation of impact in risk assessments. The impact may be on economy, human life, physical infrastructure, business, etc.
(5) **Describe communication mechanisms.** The communication capabilities for each component are described, looking into which communication standards are used, e.g., WiFi, Bluetooth, LoRa6.
(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 as described by the SCM.
(8) **Determine the protection level.** The user identifies relevant protection criteria for each component together with the respective security functionalities. These are compared to the Protection Level table given by the SCM (see also Fig. 3 on page 12).
(9) **Determine the exposure level.** Use the information from the previous two steps in the lookup Table 1.
(10) **Determine the security class.** Using the exposure and impact levels apply the lookup Table 2.

Working with the SC methodology is manual, as far as the research papers [29, 31] describe it. Therefore, the above process is also manual, with the advantage that a clear procedure is given to the user to follow. One can easily see that some of the above steps can be more or less automated. Automation is a highly desired method for making a difficult technical process more user friendly, as it reduces the number of tasks the user has to do. Steps 1 to 3 are manual, and the user can take as much time and space for writing down the descriptions 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. This is also manual and requires security expertise. 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. Steps 9 and 10 are mechanical tasks, done through lookup tables.

As such, 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 was done over several workshops (each of 30min to 1h) interacting with the SCOTT users only. The major activity during this stage was to apply the SCM ten-steps to the pilots from SCOTT, together with the respective partners.

We had *two goals*:

(1) The SCOTT users to understand how the SCM works and how to use it to apply it themselves.
(2) Us to understand how easy it is to apply the ten-steps process to the IoT systems of the SCOTT pilots that we chose as test cases.

For both goals, our interactions were geared towards collecting information about the usability of the ten-steps and how to improve it to fit the two examples that we considered representative of the intended application area.

The participants were the two teams that were working on the two SCOTT pilots detailed below. During each workshop we had between two and four persons, where one was in management position (from the coordinating team of the respective pilot) and had broad knowledge about the respective system and the others were technical people closely involved in the developing team (e.g., from GUT, Tellu IoT, AVL). These two teams of users have continued to interact with us until the last stage and the high-fidelity prototype.

We performed our studies on two applications:

(1) The "Elderly UI" component of the "Assisted Living and Community Care System" (ALCCS) pilot, coordinated by Philips Research. In short, the Elderly UI (see Fig. 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 is able to continuously observe activity and position from the elderly resident, and periodically transmits the observations straight to the Cloud.

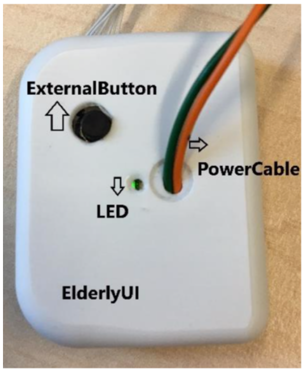


Fig. 2. Early prototype of the ElderlyUI component.
(Description and image, courtesy of Philips Research.)

(2) The "Multimodal Positioning System" (MPS) component of the "Secure Connected Facilities Management" pilot, coordinated by Vemco. The MPS had as main functionality the localisation of people and assets within critical infrastructures, being applied in this case inside a refinery.

We ran two workshops for each test case. During these, the ten-steps process went through two major redesigns, where mainly the order and number of the steps were changed, and the helping descriptions were improved.

In each workshop we used the co-discovery technique [21, 24], which is especially useful in such an early design phase, with discussions going between us, the research team, and the respective SCOTT team. Thus, during these workshops, we adjusted our understanding of the test systems and worked with the teams to understand how to properly apply the SCM to their systems. As materials, besides the previous presentations, we also used the technical project-internal documents for each test system to collect the necessary information for evaluating the connectivity, protection, and exposure levels.

Each workshop also employed the active intervention technique, which is excellent in discovering a wealth of diagnostic information about the prototype [11], which in our case was the ten-steps process. We were guiding the SCOTT team, meaning 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.

One playful activity that our users enjoyed 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 to the system. This is one major benefit claimed by the main article [31] of the security classification methodology. Therefore, our interactions 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 Major findings. Besides the constant feedback that we received during the workshops about small improvements to the ten-steps process, we made the following major observations.

(1) The participants could now perform most of the ten steps, under our guidance.
(2) The most difficult parts of the methodology were identified as being:
   (a) the evaluation of the Impact level, which, they said: "Looks like the job of a security expert" (which the participants were not); and
   (b) finding the Protection level, since it involved answering many specific security questions that needed interactions with other members of their development teams.

Explanations and Recommendations. Regarding the observation 2a, the SCM papers [30, 31] 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. Recall from the Introduction that our goal in this work 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. Therefore, since we are taking the SCM as given, and do not aim to improve it as a security instrument per se, we decided that based on observation 2a we would only improve in the future tool the help text for this step by explaining the above aspect to the users so that they can get over this step with less concern.

The second observation 2b is, however, 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 [30, 31]. Therefore, we decided to improve on how the users work with this list in the next iteration.

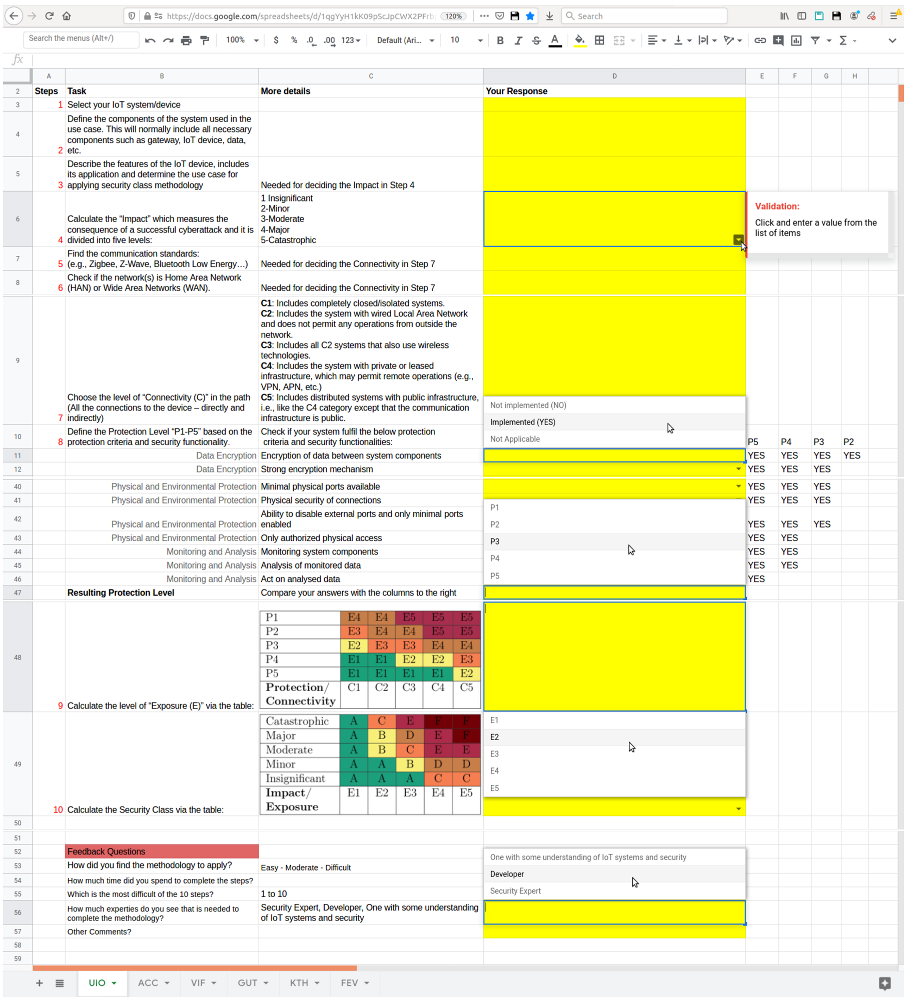
5 CREATING THE SC TOOL

In this section we describe how we built an online tool implementing the ten-steps process, and how we tested it with users during multiple testing sessions for different prototypes. To further consolidate the cognition support, we followed the computational offloading principle [28] and built a tool to help the user with their tasks by organizing, guiding, and automating some of the aspects of the task. In our case, the task at hand was the process of security classification, which also had some of the steps ready for automation; whereas for the other steps the tool was intended to help with organizing the work and gather inputs from the users.

5.1 Spreadsheet implementation

Our first low fidelity prototype was in form of a spreadsheet and was implemented in Google Sheets because, as a cloud application, it allows a team to collaborate in real-time.

The spreadsheet template (see Fig. 3) contained all the information from the previous ten-steps process, albeit in a more structured way, having the following components:



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

	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:						
49	10	Calculate the Security Class via the table:						
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								

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

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

Step: The step number coordinates the attention of the user and helps direct the workflow.
Task: A column providing the task description; the text is adopted from the ten-steps described before.
More details: This column provides additional descriptions to make the task easier to understand.

Your Response: This column stores the input from the user to the respective task, collected in three ways:

– **Free Text:** The users could freely describe the system with components and relevant functionality.

– **Dropdown list:** For inputs predefined by the methodology, requiring a specific item form a list (e.g., connectivity, protection level, presence of security functionality), we also applied a validation mechanisms to guide the users.

– **Lookup table:** The respective lookup table was given for deciding the exposure and security class.

Protection level requirements: Additional information columns displayed protection level requirements, guiding the users to compare and select the appropriate protection level (see line 47 in Fig. 3).

Our goals were to simplify the security classification process and to present and test it with more users. Therefore, for this low-fidelity prototype, we focused on providing clarifying text and necessary helper information for each step, based on the observations from Sec. 4.3.2.

5.1.1 Performing the studies. The participants were SCOTT partners (seven teams, including the two from Sec. 4.3.1) and the same students from before, as detailed further. The participants included security experts, developers and system managers having a general understanding of the IoT system and security.

One pilot test was carried with AVL, one of the SCOTT partners. The result is the one presented in Fig. 3 and is the one that we have used to do our final webinar tests.

We organised a webinar for the whole SCOTT project partners, where the two teams from Sec. 4.3.1 also helped us with the organization. We used classical methods of advertising to attract many participants, like preparing an invitation text presenting the webinar (similar to how one would do for an academic event or hackathon, but more flashy) and e-mailing it to everyone in the project, with reminders, etc. We worked with the project coordinators to make the invitation text interesting for our audience, i.e., many of the SCOTT partners were companies.

The plan for the webinar was:

(1) An introduction from us, which included: (i) a motivating presentation of the SC methodology, taken from the research papers, (ii) a short presentation of the ten-steps process, (iii) with an exemplification of how we used the ten-steps on the application from Sec. 4.3.1(1), meant as additional motivation and inspiration for the participants since it was from the same project.
(2) A hands-on interaction from the participants with the online spreadsheet.
(3) A brief (since we were restricted by the time availability of our participants) questionnaire at the end of the spreadsheet (see bottom of Figure 3).

The 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 (based on the SCOTT pilot that they were working on) and took our hands-on exercise. Each team (see bottom of Fig. 3) had to fill in our spreadsheet template according to their IoT system of choice. The exercise took between 7-30min to complete.

For part (2) we used direct, unobtrusive observation, where we were observing online how the teams were progressing. This was possible due to the capabilities of the Google Sheets to show the changes done by the participants, synchronously and in real-time. At times we had to answer questions, usually for clarification or confirmation.

One final workshop was done with the students, using a very similar setup and activity as above, during one hour of their exercise classes, i.e., we presented the spreadsheet tool and asked them to apply it to the same system as before, under our observation this time.

5.1.2 *Major findings.*

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

User help/manual: Even if the spreadsheet and terminologies were explained in our presentation, all users still had questions either for clarifying individual steps or how to assign 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 did not allow to change the lookup tables, which participants observed as a necessity when changing the type of system.

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, again, the evaluation of *impact* and the *protection level* calculation.

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.

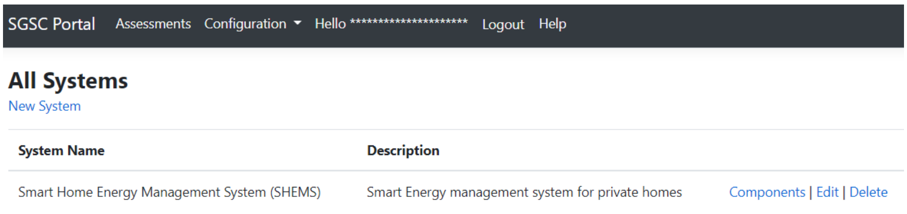
The student workshop confirmed that the ten-steps were now considerably easier to use than in the previous session when only the research papers were given.

5.2 Web-based SC tool pilot testing

The high-fidelity SCT was implemented as a web application.[22] The major technologies used were the following. The development used ASP .NET Core and the Model View Controller pattern [15], implementing also a separate service layer to provide a public RESTful API, useful when integrating in a DevOps tool-chain. We used Azure SQL database for data persistence and deployed the application in Microsoft Azure cloud services.

We simplified the assessment process by combining several steps into one, with main activities now being:

(1) **Define a System** (corresponding to step 1 from Sec. 4.3) with a snapshot in Fig.4.



All Systems

New System

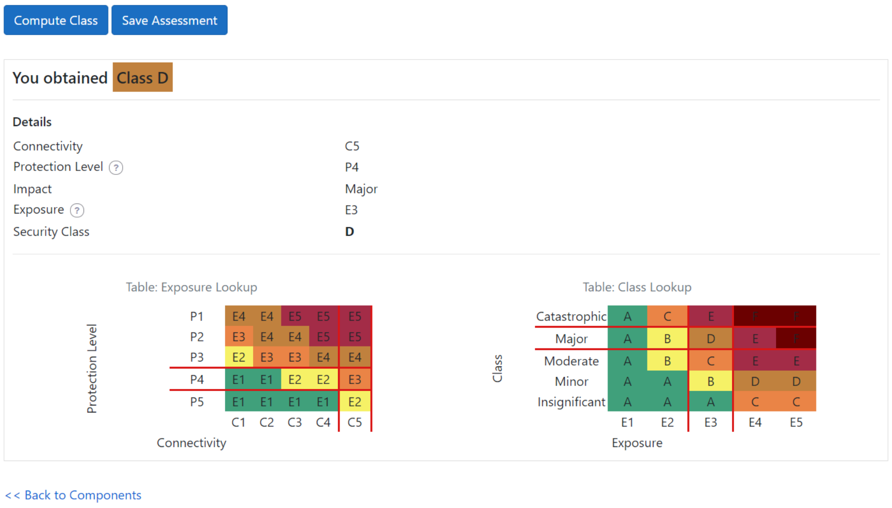
System Name	Description	
Smart Home Energy Management System (SHEMS)	Smart Energy management system for private homes	Components | Edit | Delete

Fig. 4. Snapshot of systems page of SCT web application.

(2) **Add components** (implementing steps 2–7 from Sec. 4.3). A system is decomposed into its components, and for each component in turn a class can be computed.[23] With the web tool we could add more organizational element, most importantly, components can now be categorized, providing 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 automatically some of the security functionalities as 'not applicable'.

[22]The final Security Classification Tool is available at https://lightsc.azurewebsites.net.
[23]See a video tutorial on the Help page of our tool: https://lightsc.azurewebsites.net/UserHelp/VideoTutorial

(3) **Perform assessment** (implementing step 8 from Sec. 4.3) where security functionalities are selected.
(4) **Compute security class** (automating steps 9–10 from Sec. 4.3) by pressing a button. Fig. 5 shows the final view containing also the lookup tables and what selections were made to obtain the resulting class.



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

Fig. 5. Snapshot of class calculation view.

5.2.1 *Pilot testing.*

The application was demonstrated to the SCOTT partners AVL and GUT, i.e., two of our main teams with whom we interacted several times during all this work. We had one workshop where we presented the new web application and demonstrated how to apply it to the original SHEMS example from research paper [30]. After the presentation, and during the demonstration, we had a long period of discussions with comments from the users. We did not perform full scale applications because these users already knew and had applied the ten-steps process.

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, to give them local information (the screenshots in this paper are taken from the final version where this feature was implemented).

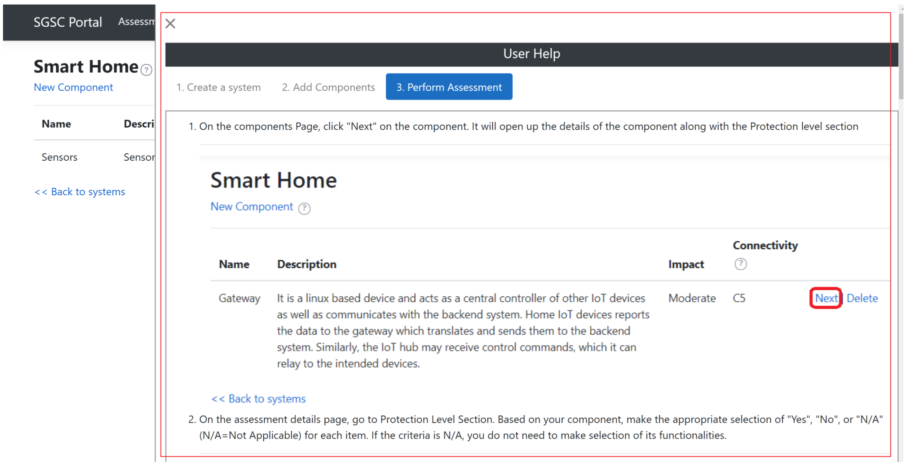
5.3 Final version of SCT

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 [30, 31]. However, as we learned from the users, depending on the domain of application, the lookup table may differ slightly. Therefore, one

should be able to change the lookup table according to their domain. The tool has a configuration feature where the user can override the default lookup table and also reset it to default.

(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. This version introduces easily, and at all times, available help menu, now being placed as a sidebar which on click, slides over the page (see Fig. 6). This sidebar allows the user to focus on their tasks, without the distraction of opening a new page each time help is needed.



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 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.

Fig. 6. Snapshot showing user help opened in a side bar from the right.

(3) **Detailed contextual help.** Since the users have constantly been asking for explanations of the terminologies and of the steps, we added help icons beside the respective texts or UI elements that required detailed explanations. When clicking on the help icon a modal window opens up to show these details. Many of these details also appear in the main help.

In this version, we also decided to implement the *“beliefs and weights”* aspect of the SC methodology from the research paper [29]. Before, in the spreadsheet, it was difficult to work with these confidence parameters; but now the tool could more easily calculate using weights and the formulas from [29], with the user only specifying the individual weights.

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 *preparations* for the hackathon included: (1) 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); as well as (3) materials for announcing and attracting participants and for managing the contest.

The hackathon event had a ca. one hour program (all 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. The *contest* asked the teams to (1) use the tool on one of their systems or components; and (2) describe how the security classes could contribute to innovation and business potential for their company. 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 from industry. The contest was thus only a framing, where our real interest resided in the *usability part* of the hackathon:

(1) We offered special recognition prizes (with extra winnings and the title of "Usability Wizards") for those that take substantial effort to help us with the usability studies, i.e., to use the two aspects mentioned below.
(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. 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 Hotjar[24] to track and analyze users' activities (i.e., interaction logs) while they were performing their evaluations with our SCT. This method of indirect observation was necessary because our participants needed the flexibility regarding doing the task that the contest asked for. We used the following particular strategies, detailed in Sec. 5.3.3: (1) *Screen recordings* of the activity of the user while working with the web tool, captured anonymously for privacy concerns.; (2) *Incoming feedbacks*, with which the users could select the specific part of the page and provide feedback on it; (3) *Heatmaps* showing which part of the page was clicked, scrolled or hovered over the most. Using this method, we were able to identify which features the users are most interested in or are most difficult and require most effort/time.

The participants attending the hackathon presentations were from four companies, of which three teams submitted the required report, with one team taking also the survey. The participation was poorer than we had expected, which was later explained by our local contact 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 already had a large client base in Poland. From the three reports that we have receive, one applied the SCM 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. They claimed that this helped them understand what security functionalities the system needed. The other two applications were to analyze the security of autonomous vehicle management systems in logistics and of RFID. Both reports used the tool similarly for trying out different security features for different configurations of their systems resulting in different security classes.

5.3.2 Evaluation with Individuals. Besides teams, we wanted to evaluate also with individuals, and thus we asked feedback from software professionals. This is the last group of users described in Sec. 3. We selected technically sound individuals and experts in software development, but not necessarily in security. In particular, we wanted individuals

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

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. Four individuals took part in the evaluation, mostly employees from eSmart Systems AS and Smart Cognition AS, both of which are software companies. We tried to organize a workshop to introduce the tool, but 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, i.e.: URLs for the tool and the video tutorial presenting how to use the SCT; Presentation slides to understand the core concept of SCM and SCT (reused from the hackathon); Description of the task, saying that the evaluation is complete after they, 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, asking also to provide feedback while using the tool, if they had any.

5.3.3 Major findings. The Hotjar data from both the hackathon and the individuals were analyzed together.

Heatmaps: 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. Another highly clicked part of this page was the compute class button (5.6%), showing that users were interested in computing the class quite often, most probably because they were repeating short cycles of changing some parameters and recomputing the class. One of the least components that users interacted with was the belief and weight inputs in the assessment page, even though the help icon to explain their concept was fairly clicked.

Screen recordings showed that the majority of users used the tool as expected. They first created the account and browsed through the description and then checked the main help page. After that, they followed the instruction of creating the system and adding sub-systems. Most of the users followed 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 (leaving these as default).

Survey: The survey showed that the users were entirely new to such classification methodologies and took 30 to 100 minutes to apply it. Similarly, learning this particular tool took between 15 to 60 minutes. One of the users who had 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.

The tool was considered usable in the planning phase by most users, with the testing phase on second place, according to the results from 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 concepts of 'belief and weights' to be the most unintelligible part of the tool, which we already observed in the heatmaps and recordings. Surprisingly, three of the five responses 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, anyone 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.

Four out of five found the methodology moderately easy. However, one of the users found it difficult to apply in his system because the user considered that assessing each protection criteria is 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. Another feedback was to provide more guidance to fill in the 'belief and weights' parameters.

6 DISCUSSIONS AND LIMITATIONS

The observations about the final version of the SC tool generally suggest that the tool is easy enough to be used by non-security experts. This encouraged us to release it as a public tool (see link on page 14). The more experimental 'beliefs and weights' part of the tool (which we purposely did not detail here) was considered not so easy. This only confirms the SCM research papers, who also considered this a complex feature.

In total, throughout all our stages of creating the tool, we saw the SCM applied to ca. 17 different IoT systems, 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 SC tool prototypes that we have been building, but can also be seen as useful proofs of the applicability of the original security classification methodology that we have worked with.

The principles for a DevOps-ready Security Classification from Sec. 2.2 have motivated our work. We have implemented the chosen methodology into a tool (following the external cognition approach), thus answering to 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), to answer Principle 3. We did not strive much in the direction of Principle 1 because, having now a tool, one can do re-evaluations of the system by making the necessary changes in the evaluation parameters and re-running the class calculation. Since our tool can provide an API, we believe that Principle 1 (dynamicity) can easily be attained; however, this is more of an engineering task that is best left to a software development company to undertake. We leave this as further work, to be done by companies willing to take up our SC tool, or similar ones, into their DevSecOps tool-chains, since the adjustments and implementations are routine.

A general recipe was thus discovered, for going from a research effort security classification methodology to a DevOps-ready tool. Any such endeavor, 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 (especially cloud-based that also offer real-time and collaborative features).
(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. At the same time, choosing well the target group representatives to include both individuals and teams, with diverse expertise, is essential for usability testing in all three phases.

7 CONCLUSION AND RELATED WORK

We have identified 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 our work is concerned with exemplifying the process of taking an existing security classification methodology and working with it towards satisfying the five principles. To do this, we have created a tool that implements the chosen methodology (thus conforming to Principle 2) and 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. 17 IoT systems, during ca. 14 workshops and larger events, involving as test users both teams and individuals over a period of 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 of 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 that was used for such evaluations before. Companies can now use existing internal resources (i.e., their developers or CTOs) for evaluating 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

We are not aware of security classification methods (or alike) that can be used within DevSecOps. Moreover, we have no knowledge of other usability studies as the one we did here, where a security methodology (of any kind) would be transformed into a tool using an interaction design process; let alone works that also identify principles and recipes for doing such an activity, as we did.

The most relevant related works can be found among existing tools that are used to support existing security methodologies. We will evaluate these here, since other forms of related works that look at alternative classification or security evaluation methods can be found in the respective references to the security classification methodology that we have chosen [29–31].

There are several tools [26] to support security experts to structure their security/safety arguments based on diagrammatic notations s.a. the Goal Structuring Notation (GSN) [32] or Toulmin's argument model [33]. NOR-STA[25] is an argumentation tool, based on [33], to support compliance, assurance and security cases [7] using Dempster-Shafer theory for aggregation of confidence parameters (i.e., the 'belief and weights' that our final SCT implements, but which we glossed over with the purpose of simplifying the presentation). The tool is sophisticated and has many features; however, it seems limited to strict predefined requirements, thus not appealing for DevSecOps. Moreover, we have not found usability studies done for this tool, and security experts seem to be the only target group. CertWare is an open-source Eclipse plugin from NASA [3] for development of safety, assurance and dependability cases that seems to be superseded by AdvoCATE [9, 10], which provides some automation support and has been applied to real systems s.a. unmanned aircraft. These last two tools work similarly to NOR-STA, are aimed specifically at security experts, and we could not find usability evaluations.

For risk assessments, STRIDE is a popular model (and tool[26]) from Microsoft for threat modelling. In the same category, CORAS is a heavy process that requires security experts and stakeholders to work together to identify threats and risks [14]. CORAS comes with a tool that uses several graphical notations, and has been applied in several real systems. ArgueSecure[27] is a recent graphical qualitative risk assessment and security requirement elicitation framework [18, 19] that is more light-weight than the above and uses an argumentation model. The authors have performed

[25]https://nor-sta.eu/en
[26]https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling
[27]https://danionita.github.io/ArgueSecure/

usability evaluations, but the tool is rather manual and meant for the security experts. Being also attack-centric, we cannot consider this tool DevOps-ready.

REFERENCES

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

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

[3] Matthew R. Barry. 2011. CertWare: A workbench for safety case production and analysis. In *2011 Aerospace conference*. IEEE, 1–10. https://doi.org/10.1109/AERO.2011.5747648

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

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

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

[7] Lukasz Cyra and Janusz Gorski. 2011. Support for argument structures review and assessment. *Reliability Eng. & System Safety* 96, 1 (2011), 26–37. https://doi.org/10.1016/j.ress.2010.06.027

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

[9] Ewen Denney and Ganesh Pai. 2018. Tool support for assurance case development. *Automated Software Engineering* 25, 3 (2018), 435–499.

[10] Ewen Denney, Ganesh Pai, and Josef Pohl. 2012. AdvoCATE: An Assurance Case Automation Toolset. In *Computer Safety, Reliability, and Security (Lecture Notes in Computer Science, Vol. 7613)*, Frank Ortmeier and Peter Daniel (Eds.). Springer Berlin Heidelberg, 8–21. https://doi.org/10.1007/978-3-642-33675-1_2

[11] Joseph S. Dumas and Janice C. Redish. 1999. *A practical guide to usability testing.* Intellect.

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

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

[14] Rune Fredriksen, Monica Kristiansen, Bjørn Axel Gran, Ketil Stølen, Tom Arthur Opperud, and Theodosis Dimitrakos. 2002. The CORAS framework for a model-based risk management process. In *International Conference on Computer Safety, Reliability, and Security*, Stuart Anderson, Massimo Felici, and Sandro Bologna (Eds.). Springer, 94–105. https://doi.org/10.1007/3-540-45732-1_11

[15] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

[16] Tony Hsiang-Chih Hsu. 2018. *Hands-On Security in DevOps: Ensure continuous security, deployment, and delivery with DevSecOps.* Packt Publishing.

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

[18] Dan Ionita, Margaret Ford, Alexandr Vasenev, and Roel Wieringa. 2018. Graphical Modeling of Security Arguments: Current State and Future Directions. In *Graphical Models for Security (Lecture Notes in Computer Science, Vol. 10744)*, Peng Liu, Sjouke Mauw, and Ketil Stolen (Eds.). Springer, 1–16.

[19] Dan Ionita, Roeland Kegel, Andrei Baltuta, and Roel Wieringa. 2016. ArgueSecure: Out-of-the-box security risk assessment. In *24th International Requirements Engineering Conference Workshops*. IEEE, 74–79. https://doi.org/10.1109/REW.2016.027

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

[21] Sue Kennedy. 1989. Using video in the BNR usability lab. *ACM SIGCHI Bulletin* 21, 2 (1989), 92–95.

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

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

[24] Kai H. Lim, Lawrence M. Ward, and Izak Benbasat. 1997. An empirical study of computer system learning: Comparison of co-discovery and self-discovery methods. *Information Systems Research* 8, 3 (1997), 254–272.

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

[26] Mike Maksimov, Nick L.S. Fung, Sahar Kokaly, and Marsha Chechik. 2018. Two decades of assurance case tools: a survey. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 49–59.

[27] Jason R.C. Nurse, Sadie Creese, and David De Roure. 2017. Security risk assessment in Internet of Things systems. *IT professional* 19, 5 (2017), 20–26.

[28] Mike Scaife and Yvonne Rogers. 1996. External cognition: how do graphical representations work? *International journal of human-computer studies* 45, 2 (1996), 185–213.

[29] Manish Shrestha, Christian Johansen, and Josef Noll. 2020. Building Confidence using Beliefs and Arguments in Security Class Evaluations for IoT. In *5th International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 244–249. https://doi.org/10.1109/FMEC49853.2020.9144957

[30] Manish Shrestha, Christian Johansen, and Josef Noll. 2020. Criteria for Security Classification of Smart Home Energy Management Systems. In *Advances in Smart Technologies Applications and Case Studies*. Springer. https://doi.org/10.1007/978-3-030-53187-4_19

[31] Manish Shrestha, Christian Johansen, Josef Noll, and Davide Roverso. 2020. A Methodology for Security Classification applied to Smart Grid Infrastructures. *International Journal of Critical Infrastructure Protection* 28 (2020), 100342. https://doi.org/10.1016/j.ijcip.2020.100342

[32] John Spriggs. 2012. *GSN-The Goal Structuring Notation: A Structured Approach to Presenting Arguments.* Springer Science & Business Media. https://doi.org/10.1007/978-1-4471-2312-5

[33] Stephen E. Toulmin. 2003. *The uses of argument.* Cambridge university press. https://doi.org/10.1017/CBO9780511840005

[34] Zhi-Kai Zhang, Michael Cheng Yi Cho, and Shiuhpyng Shieh. 2015. Emerging Security Threats and Countermeasures in IoT. In *10th ACM Symposium on Information, Computer and Communications Security* (Singapore, Republic of Singapore) *(ASIA CCS '15)*. ACM, 1–6. https://doi.org/10.1145/2714576.2737091

UiO : Department of Informatics
University of Oslo

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

157

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.

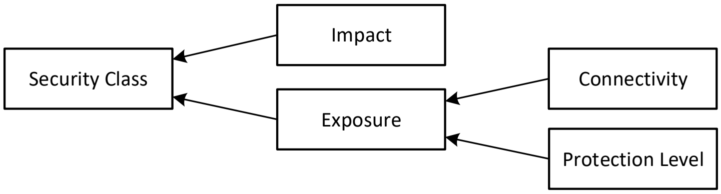


Figure 1: Components of the evaluation of a security class.

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” (ALCCS) 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;

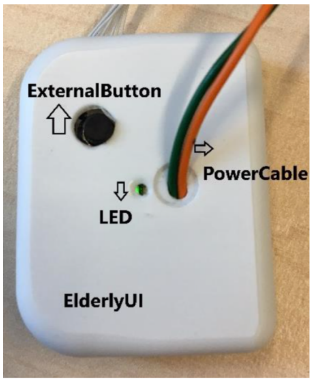


Figure 2: Early prototype of the ElderlyUI component.
(Description and image courtesy of Philips Research.)

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.

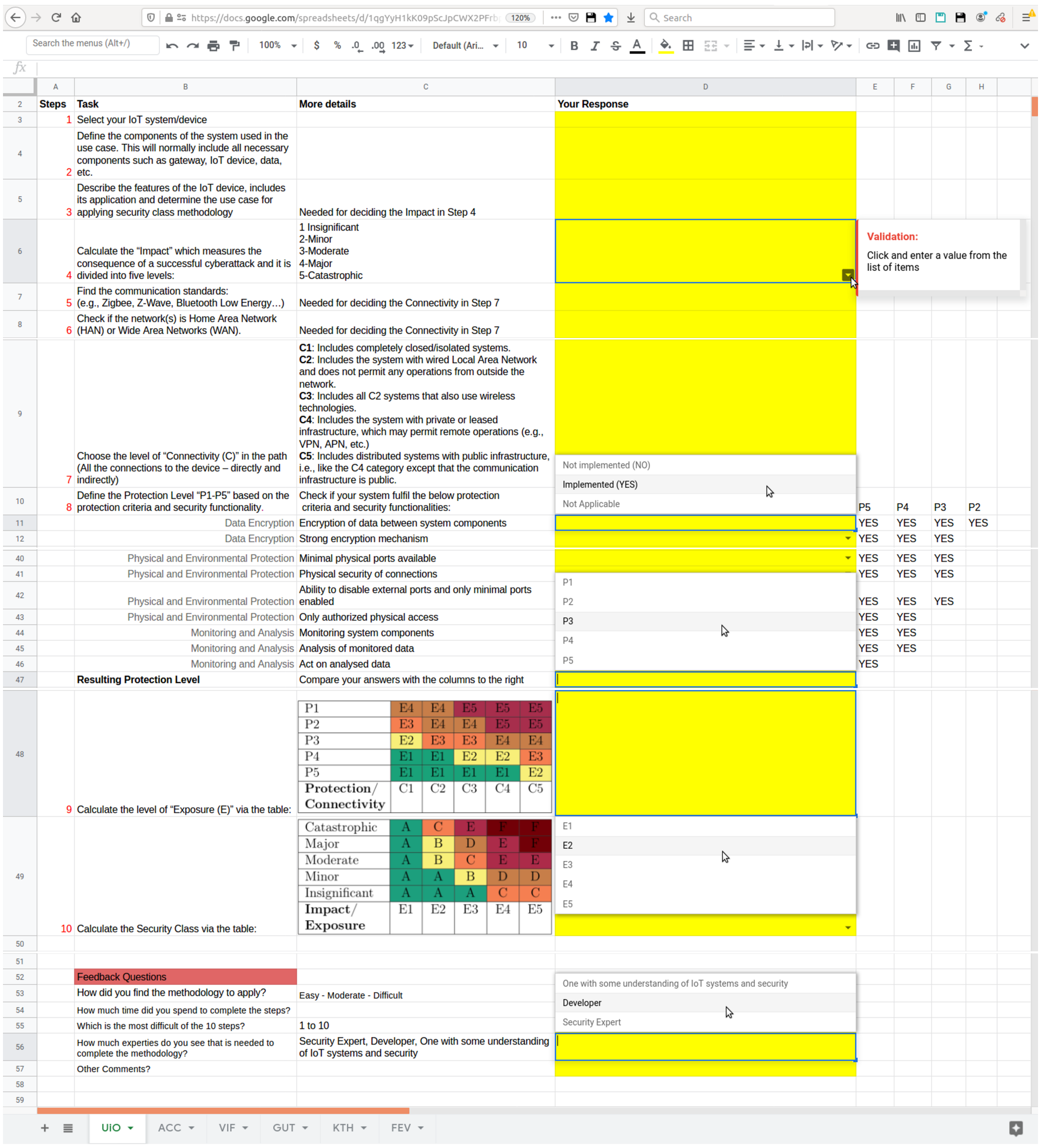
Lookup table: The lookup table was also shown to guide the user to provide valid exposure and security class.

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:						
49	10	Calculate the Security Class via the table:						
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								

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

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

Not implemented (NO)
Implemented (YES)
Not Applicable

P1
P2
P3
P4
P5

E1
E2
E3
E4
E5

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

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

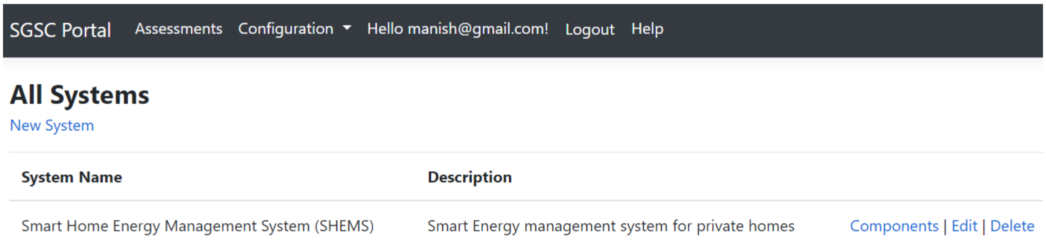
[26]https://azure.microsoft.com/

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.



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.

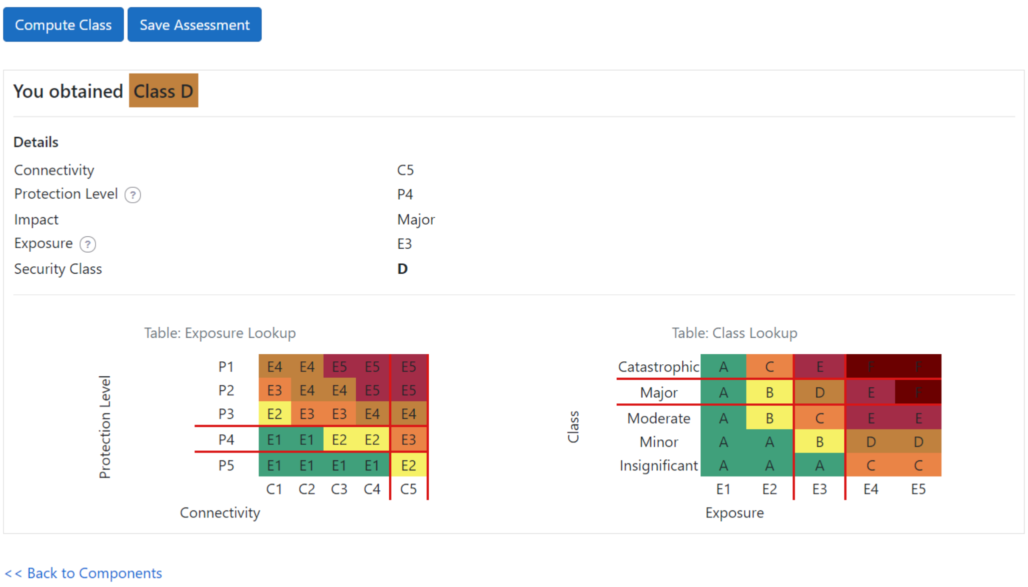
3. **Perform assessment** (corresponding to step 8 from Section 4.3)

   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					
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	C1	C2	C3	C4	C5

Table: Class Lookup

Class					
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	E1	E2	E3	E4	E5

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.

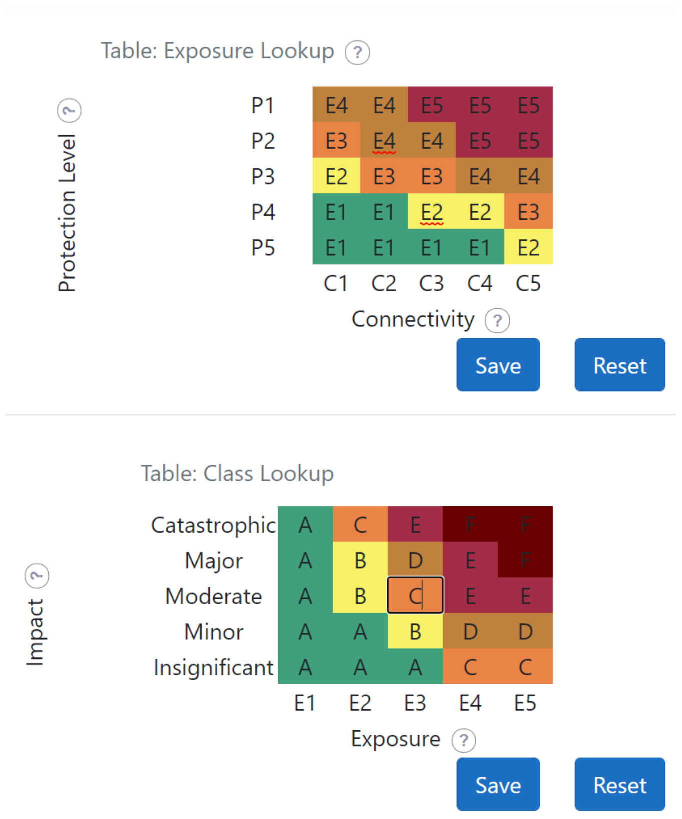


Table: Exposure Lookup

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

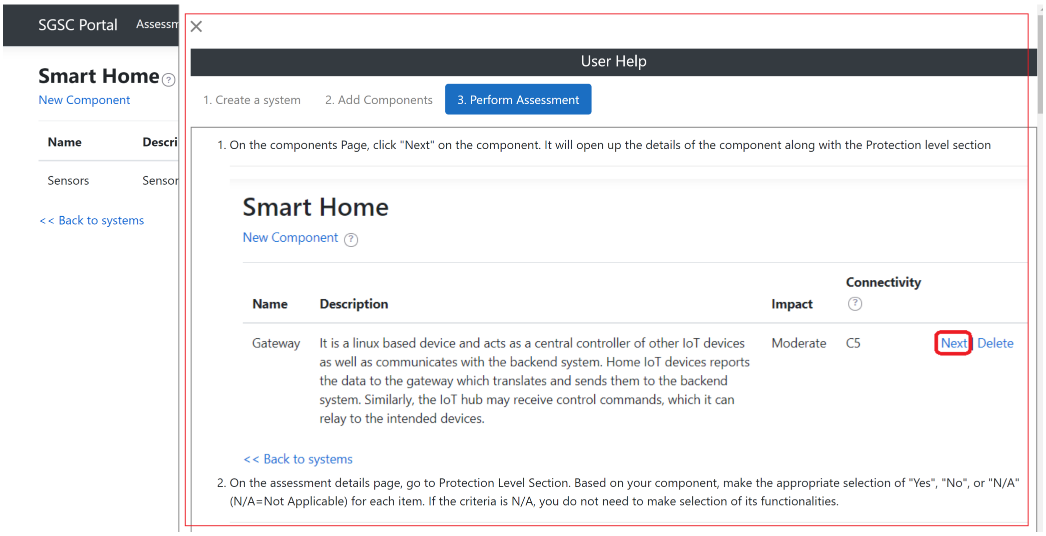
Figure 6: Snapshot showing the customisation of lookup tables.

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.



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:

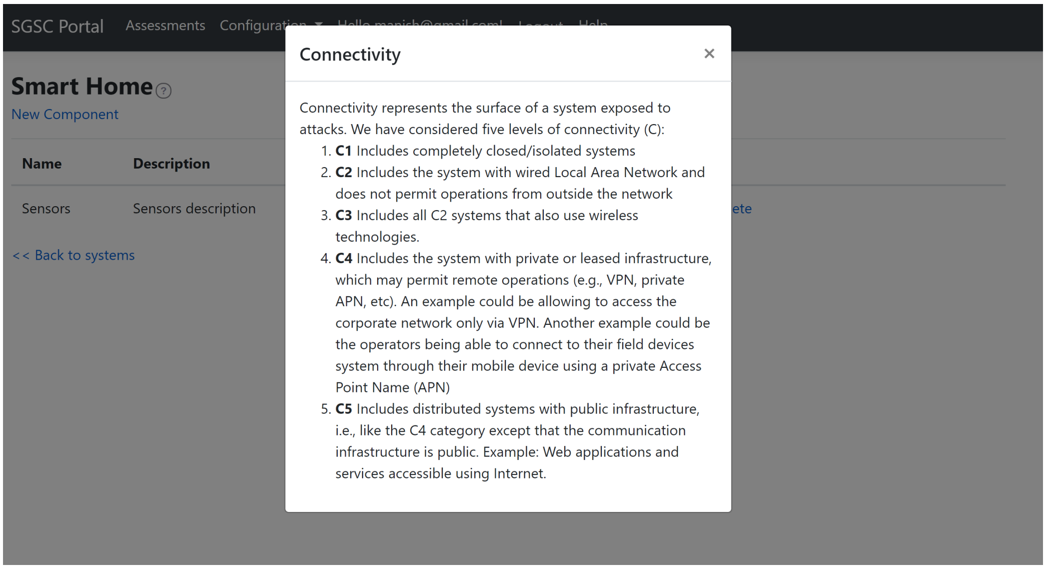
1. preparing a video tutorial (ca. 10min) on how to use the tool;
2. preparing a presentation with slides
   (a) motivating the concept of security classification,
   (b) describing the benefits for industry,
   (c) explaining the ten-steps process, and
   (d) how to apply it to the SCOTT pilot (this 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.

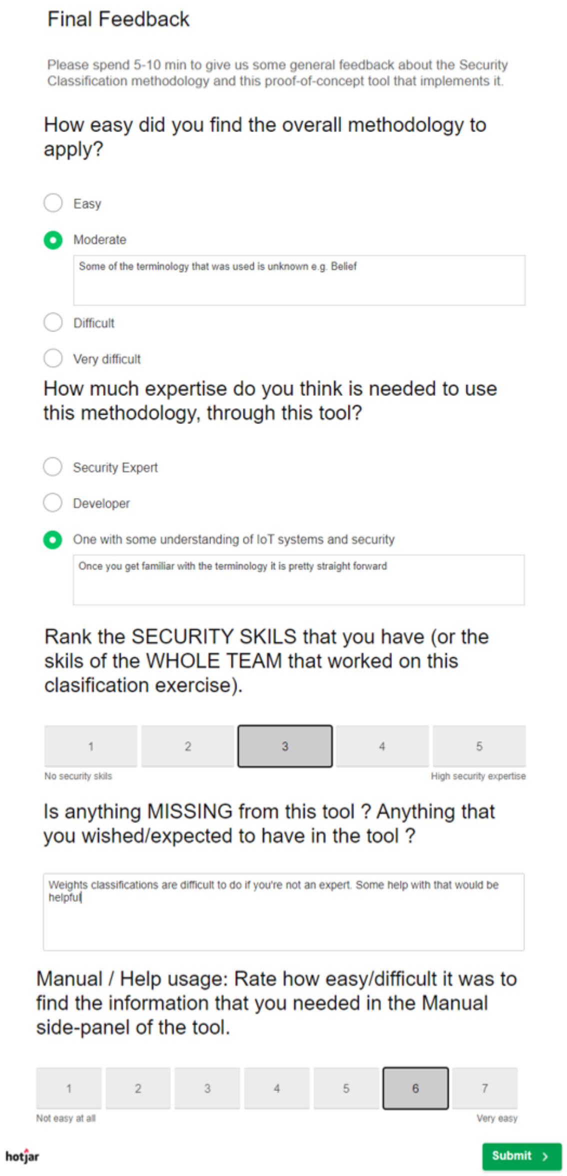
[20] Manish Shrestha, Christian Johansen, and Josef Noll. Criteria for Security Classification of Smart Home Energy Management Systems. In *Int. Conf. Smart Information & Comm. Technologies.* Springer, 2019.

[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]
  Some of the terminology that was used is unknown e.g. Belief
- Difficult
- Very difficult

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 ?

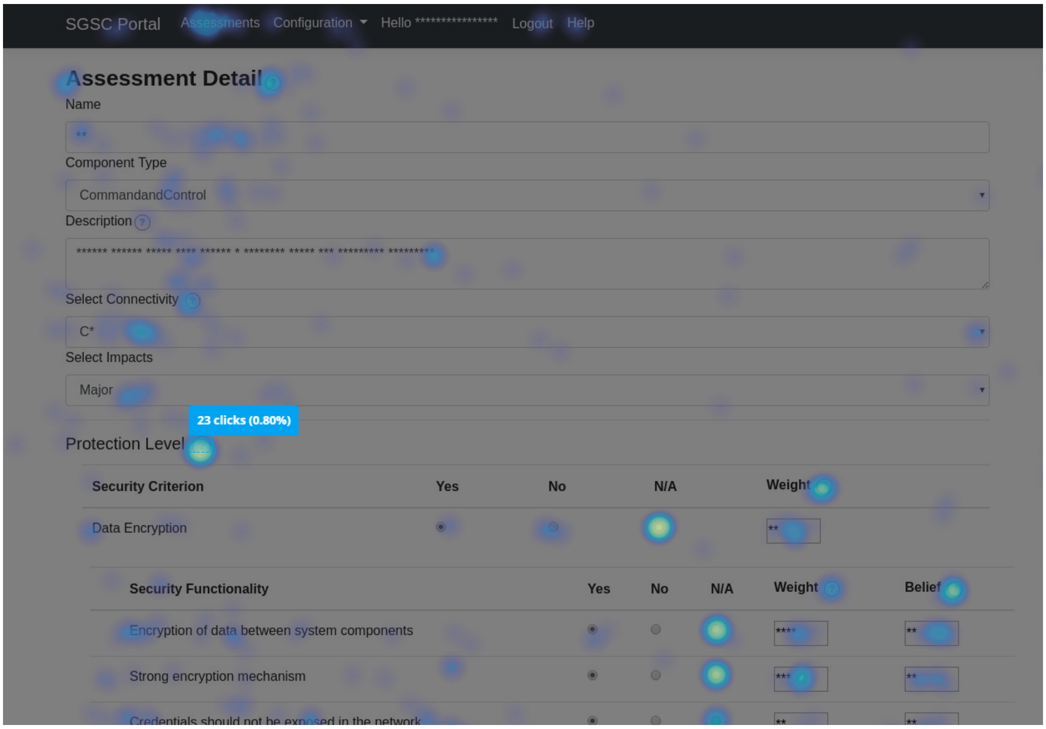
Weights classifications are difficult to do if you're not an expert. Some help with that would be helpful

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*

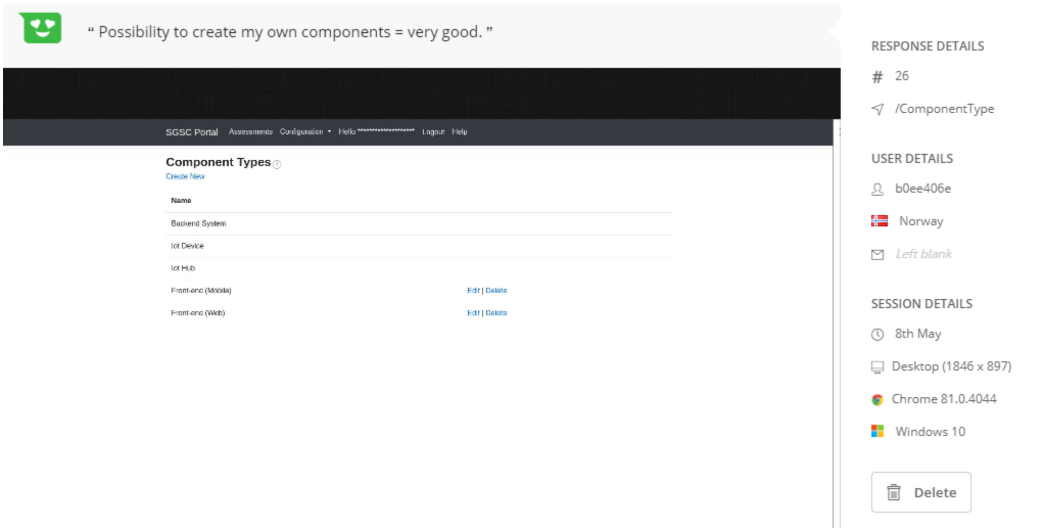
Select Impacts: Major

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
Create New

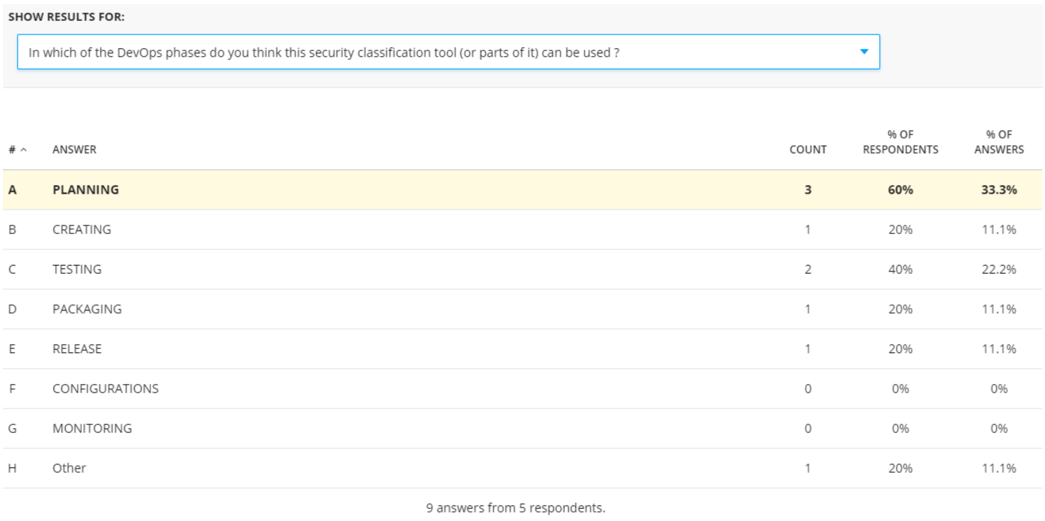
Name	
Backend System	
Iot Device	
Iot Hub	
Front-end (Mobile)	Edit | Delete
Front-end (Web)	Edit | Delete

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.