EUROPEAN UNION AGENCY
FOR CYBERSECURITY

ADVANCING SOFTWARE SECURITY IN THE EU

The role of the EU cybersecurity certification framework

NOVEMBER 2019

ABOUT ENISA

The mission of the European Union Agency for Cybersecurity (ENISA) is to achieve a high common level of cybersecurity across the Union, by actively supporting Member States, Union institutions, bodies, offices and agencies in improving cybersecurity. We contribute to policy development and implementation, support capacity building and preparedness, facilitate operational cooperation at Union level, enhance the trustworthiness of ICT products, services and processes by rolling out cybersecurity certification schemes, enable knowledge sharing, research, innovation and awareness building, whilst developing cross-border communities. Our goal is to strengthen trust in the connected economy, boost resilience of the Union's infrastructure and services and keep our society cyber secure. More information about ENISA and its work can be found at www.enisa.europa.eu.

CONTACT

For contacting the authors please use isdp@enisa.europa.eu.
For media enquiries about this paper, please use press@enisa.europa.eu.

CONTRIBUTORS

Rob van der Veer (SIG), Menno Valkema (SIG), Fabio Guasconi (Bl4ckswan), Prokopios Drogkaris (ENISA)

EDITORS

Prokopios Drogkaris (ENISA) and Athena Bourka (ENISA)

ACKNOWLEDGEMENTS

We would like to thank Apostolos Malatras (ENISA), Philippe Blot (ENISA) and Eric Vetillard (ENISA) for their valuable comments and reviews while preparing this study.

LEGAL NOTICE

Notice must be taken that this publication represents the views and interpretations of ENISA, unless stated otherwise. This publication should not be construed to be a legal action of ENISA or the ENISA bodies unless adopted pursuant to the Regulation (EU) No 2019/881.
This publication does not necessarily represent state-of the-art and ENISA may update it from time to time.
Third-party sources are quoted as appropriate. ENISA is not responsible for the content of the external sources including external websites referenced in this publication.

This publication is intended for information purposes only. It must be accessible free of charge. Neither ENISA nor any person acting on its behalf is responsible for the use that might be made of the information contained in this publication.

COPYRIGHT NOTICE

© European Union Agency for Cybersecurity (ENISA), 2020
Reproduction is authorised provided the source is acknowledged.

For any use or reproduction of photos or other material that is not under the ENISA copyright, permission must be sought directly from the copyright holders.
ISBN 978-92-9204-343-8, DOI 10.2824/81950

TABLE OF CONTENTS

1. INTRODUCTION 4

1.1 SCOPE – OBJECTIVES 4

1.2 STRUCTURE OF THE DOCUMENT 5

2. TOWARDS IMPROVED SOFTWARE SECURITY 6

2.1 SECURITY REQUIREMENTS 6

2.2 SECURE SOFTWARE ENGINEERING 6

2.3 SECURE DEVELOPMENT LIFECYCLE 7

2.4 EXISTING STANDARDS AND GOOD PRACTICES 7

3. WHAT IS LACKING IN SOFTWARE SECURITY 10

3.1 CLEAR GUIDANCE 10

3.2 QUALITY ASSURANCE 10

3.3 SUSTAINED TRUST ISSUES 11

3.4 ASSURANCE CLARITY BETWEEN PROCESS AND PRODUCT 11

4. DRIVING SOFTWARE SECURITY IN THE EU 12

4.1 DEVELOP A COMMON REPOSITORY FOR SHARED SECURITY MEASURES 12

4.2 IMPROVE TECHNICAL STANDARDS LANDSCAPE 12

4.3 PROVIDE ASSURANCE IN THE ENGINEERING PROCESS 12

4.4 INCREASED SCHEMES APPLICABILITY AND CLEARLY COMMUNICATED ASSUMPTIONS 13

5. REFERENCES 14

EXECUTIVE SUMMARY

Secure software development and maintenance is attracting a lot of attention lately, due to the rapidly increased dependency of everyday products, services and process to the underlying software. Quite often, weaknesses behind reported security incidents and/or breaches are being materialized due to the lack of adherence on fundamental security principles and techniques. In order to promote further the assurance on the level of security or even the mitigated security threats, software development and maintenance is becoming increasingly subject to evaluation, and eventually certification, of ICT products, services and processes. Based on this, as part of ENISA activities in the area of supporting the preparatory policy discussions in the area of certification of products, services and processes, this study aims to touch upon the aspects to be considered in EU cybersecurity certification schemes (relevant to software development and maintenance).

This study discusses some key elements of software security and provides a concise overview of the most relevant existing approaches and standards while identifying shortcomings associated with the secure software development landscape, related to different inherent aspects of the process. Lastly, it provides a number of practical considerations relevant to the different aspects of software development within the newly established EU cybersecurity certification framework and the EU cybersecurity certification schemes. These considerations are listed below:

- Manufacturer(s) or provider(s) of certified ICT products, ICT services or ICT processes, should explore the deployment and maintenance of repositories not only for publicly disclosed vulnerabilities but also for shared security aspects of certified products, services and processes towards aligning on requirement commonalities and ways to mitigate common security risks.
- Following the publication of the Union Rolling Work Programme, European Standards Organizations (ESOs) and Standards Developing Organization (SDOs) should coordinate on the priority areas they can support, put forward standardization activities to benefit the future developed schemes and communicate periodically such planning to the EC and relevant CSA stakeholders.
- EU cybersecurity certification schemes for products, services and process should include, to the extent possible, not only requirements for the end product/service/process but also assurance for the engineering process, by setting process guidelines for software development, maintenance and operation.
- During the development of EU cybersecurity certification schemes, lightweight conformity assessment methods for the basic assurance level should be considered as a response to the existing fragmented landscape of software development and maintenance.
- Software developers and product manufacturers should put forward their experience and expertise and promote the uptake of EU cybersecurity certification schemes.

1. INTRODUCTION

Software is

This study is envisioned as a reference document that complements similar ongoing initiatives at National level, while drafting candidate cybersecurity certification schemes and also as a non-binding guidance document for EU cybersecurity certification framework stakeholders during the promulgation and maintenance of adopted EU cybersecurity certification schemes.

A more detailed analysis in the area of IoT security, with a particular focus on software development guidelines for secure IoT products and services throughout their lifetime, is also available by ENISA [7]. The focus of the aforementioned publication was to define a set of good practices and guidelines to be applied in the different phases of the secure SDLC of IoT solutions.

1.2 STRUCTURE OF THE DOCUMENT

The rest of the document is structured as follows:

- Section 2 discusses some key elements of software security in order to allow a better comprehension of the document's direction, being completed with an overview of the most relevant existing approaches and standards.
- Section 3 provides an overview of the shortcomings inside the software security landscape, related with different inherent aspects of the process, of the products and of the concepts surrounding the software security concept itself.
- Section 4 provides a number of practical considerations that can be considered and adopted with regards to the different aspects of software development within the newly established EU cybersecurity certification framework and the EU cybersecurity certification schemes.

2. TOWARDS IMPROVED SOFTWARE SECURITY

2.1 SECURITY REQUIREMENTS

Functional requirements are about behaviour of the system towards the outside world (e.g. a user), whereas non-functional requirements are mainly about the internal mechanisms. Many of the security requirements are non-functional; for example on how to store passwords in a database. Security requirements originate from different sources, such as

- explicit functional and non-functional requests from user(s),
- requirements and obligations originating from the underlying legal framework
- requirements that are considered as best practices, company policies, in widely accepted guidelines, from threat assessment but also, from the experience of a developer, e.g. “I always make sure my error messages don’t contain any personal information”.

In some sectors, a specific security standard and/or relevant certifications have become common practise, such as the ones produced by the Payment Card Industry Security Standards Council (PCI SSC)[2] baselining security requirements in the payment card processing domain. Compliance to PCI SSC standards is not a legal requirement (with some small exceptions), yet it is typically contractually transmitted down the chain to vendors and service providers. As such, in this area, PCI SSC standards have become the norm on duty of care and play a role in liability of service providers since not meeting the requirements could be regarded as negligence. Such implications make organisations treat PCI SSC standards almost as a de facto set of requirements. Another example is the ISA/IEC 62443[3] series of standards, which is receiving growing interest in the domain of industrial control systems.

2.2 SECURE SOFTWARE ENGINEERING

The ISO/IEC 27000:2018 [9] definition of information security mentions “preservation of confidentiality, integrity and availability of information” while noting that additional “properties such as authenticity, accountability, non-repudiation and reliability can also be involved”. Software security typically involves reducing the probability and impact of unauthorized/inappropriate access, use, disclosure, disruption, deletion/destruction, corruption, modification, inspection, recording or devaluation of data and functions managed by software. Technically, software security is achieved with an ongoing secure software development and deployment process, or secure software engineering where security is built in, and provided, involving people, tools and practices.

Secure software engineering can be seen as providing countermeasures against applicable security threats. Therefore it is important to identify the necessary mitigating measures, deploy them properly and verify their implementation as a safety net for unwillingly introduced vulnerabilities. Verification can be performed through automated or manual tests and manual review. Therefore secure software engineering is not just about finding security weaknesses but also about preventing the mistakes that cause them, through training, instruction, technology choices, security by design, etc. In that sense, secure software engineering can be seen as an important part of the practice of preventing and mitigating security risks.

[2] https://www.pcisecuritystandards.org/
[3] https://www.isa.org/intech/201810standards/

Key elements of secure software engineering are: threat modelling, risk analysis, guidelines, education, requirements, design analysis, issue management, compliance, verification, defect management

hardening threat modelling 1) requirements, 2) compliance, 3) education, 4) guidelines, 5) risk analysis, 6), 7) defect management, 8) verification, 9) design analysis, 10) issue management and 11) hardening. Software maintenance is typically defined as the engineering phase that happens after the software is released. On average, this phase accounts for 80% of the total development effort [10], which means that software quality aspects during maintenance should remain important attention points. Therefore, within the scope of this document, software development includes maintenance.

To this respect, Security Maturity Models (SMM) are a very useful tool since they guide organisations in defining their level of security in accordance with the requirements they wish to fulfil [7]. The maturity of security assesses the understanding of the current level of security, its needs, benefits, and the cost of its support. This assessment takes into account specific threats to the regulatory and compliance requirements of an organisation's sector, the unique risks present in an environment, and the organisation's threat profile.

2.3 SECURE DEVELOPMENT LIFECYCLE

The Secure Development Lifecycle (SDL) is the introduction of security activities into processes involved in application management, application provisioning and operation, infrastructure management and application audit, collectively representing the Software Development Lifecycle (SDLC) [11]. SDLC is the process of the development of a software that includes planning, analysis, design, testing, and implementation. Software developers may use different models with regards to development [12], such as Waterfall, Iterative or Agile. However, all models usually feature the following activities i) requirements elicitation, ii) software design, iii) development/implementation, iv) testing and acceptance, v) deployment and integration and vi) maintenance and disposal. Within the SDL, the principle of security by design comes into play as security should be built into the product/software from its inception and be constantly reviewed until it is ready for release to customers.

2.4 EXISTING STANDARDS AND GOOD PRACTICES

There are several standards and good practices focusing on software security. Most notably:

- Common Criteria[4] is a consolidated and widely recognized framework for product (often times meaning software) security evaluations. The evaluation process, in essence, pertains to the assessment against pre-defined security functional and assurance requirements. Depending on the assurance level to be reached (ranging from the minimum level 1 to the maximum level 7) a different set of security assurance requirements is applicable.
- The OWASP ASVS (Application Security Verification Standard)[5] is a community developed verification framework focusing on technical security controls verifiable in the software product and in the development process. It distinguishes maturity levels that reflect a system's security profile; this suggests which security requirements must or may apply. Apart from this, ASVS foresees use cases that include developer training, developer/architect reference, supplier governance, guidance on: planning, documentation, threat modeling and coding guidelines, pipeline automation, test automation and dependency management.

[4] https://www.commoncriteriaportal.org/
[5] https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

- BSIMM[6] (latest version 9, 2018) is a commercial initiative, based on a bottom-up approach that starts with listing activities of large successful software companies. This benchmark shows which controls can be considered in maturity terms, in which higher maturity levels are typically a step-up from lower maturity levels. Its four categories of controls map largely to the categories in OWASP SAMM.
- BSI PAS 754 "Software Trustworthiness. Governance and management. Specification"[7] is a standard developed by the British Standards Institution encompassing both security and reliability concerns (safety, reliability, availability, resilience and security). It describes a wide applicable approach for organizations aiming to adopt system trustworthiness practices and is based on the concepts of governance, risk management, control applications and compliance.
- ISA 99 / IEC 62443 provides a set of standards aimed at industrial control systems and provides a flexible framework to address and mitigate current and future security vulnerabilities in industrial automation and control systems (IACSs). It also includes a standard for secure software development (62443-4-1) which specifies process requirements for the secure development of products used in industrial automation and control systems. Requirements include verification of formal processes being used and visible in the product. Thereby this standard verifies process requirements partly by checking characteristics of the product, which underlines the solid relationship between the two.
- ISO/IEC 27034 is a multipart, guidance international standard focusing on application security. Each of its numerous parts goes down in deep details on how software security should be achieved. The most relevant ones are:
    - 27034-3: application security management process;
    - 27034-4: validation and verification (still to be published);
    - 27034-5: protocols and application security controls data structure;
    - 27034-7: assurance prediction framework.

  Its development has been strongly backed by large software enterprises in order to fill the gap in international standardization on this topic.
- ISO/IEC 62304 is a certifiable standard in the field of medical (device) software focusing on life cycle requirements for the development of medical software and software within medical devices. It distinguishes three classes of safety requirements which are similar to a level of assurance/criticality of system security. It is divided into separate concerns of software maintenance, (incident) response, configuration management and risk governance. The different categories follow common steps in a software development process (from design/planning to implementation/verification, release/anomaly response).
- PCI SSC has been relying for the last decade on the PA-DSS standard for payment applications certification. More than 5000 software products have been certified during this period of time relying on a structured mechanism including accreditation, demanding third-party audits and incremental certification for new versions. Currently PCI SSC is in a transition for a new framework for software security which will be in full effect in 2022 and will leverage two different standards: one addressing the SDL and one the security of the product itself.
- OWASP Software Assurance Maturity Model (SAMM)[8] is a community developed framework to help organizations formulate and implement a strategy for software security. The framework covers four core business functions of software development, namely i) governance, ii) construction, iii) verification and iv) operations, with security

[6] https://www.bsimm.com/
[7] http://tsfdn.org/wp-content/uploads/2016/03/TS502-0-TS-Essentials-Specification-Issue-1.2-WHITE.pdf
[8] https://www.owasp.org/index.php/OWASP_SAMM_Project

practices tied to each one of them. The building blocks of the model are the three maturity levels defined for each of the twelve security practices. These define a wide variety of activities in which an organization could engage to reduce security risks and increase software assurance.

- The Microsoft SDL is considered a classical model in secure development frameworks. It distinguishes Training (a prerequisite), Requirements, Design, Implementation, Verification, Release, and Response (to external or unexpected events).
- The Dutch SSA (Secure Software Alliance) has defined a framework[9] for secure software development intending to conform to all phases of the SDLC. It focuses on threat modeling as a prerequisite for secure software development.
- Safecode.org[10] is an initiative to identify and promote best practices for secure software development. Their "Fundamental Practices for Secure Software Development" follows steps in the development process from governance to design, coding, testing, and vulnerability response. Relevant to the present work, Safecode has published a short document[11] with its vision on cybersecurity certification in which they note the following:
    - Security certification should encompass the complete product (including configurations in its deployment operation), not just security features.
    - Certification should be possible on an unfinished product. Development process has more predictive value of a product's security than testing it afterwards.
    - Certification levels should be linked to product classes, to lessen the assumption that the highest level is always preferred.

Further to the certifications that might accompany some of the aforementioned standards, some lightweight certification schemes have also been developed such as the CSPN (France), the BSZ (Germany), LINCE (Spain) and BSPA (Netherlands).

[9] https://securesoftwarealliance.org/framework-secure-software/
[10] http://www.safecode.org/
[11] https://safecode.org/wp-content/uploads/2018/02/SAFECode_Perspective_on_Cybersecurity_Certification.pdf

3. WHAT IS LACKING IN SOFTWARE SECURITY

3.1 CLEAR GUIDANCE

While, as presented earlier, many software security related standards exist, their requirements largely overlap. This overlap demonstrates that software security is mainly a generic problem and both Standards Developing Organizations (SDOs) and European Standards Organizations (ESOs) or good practice producers are often working without proper coordination and effective liaisons. Further to that and as already acknowledged by relevant ENISA work in the area [13], traditional standardisation processes can be time-intensive, potentially causing delays in the application of necessary standards and interoperability. Considering the number of existing approaches, there are no widely used standardized ways for assessing horizontally the security of software products, apart from some methods to perform a penetration test or a code review. The situation is even worse if we consider software security certification as only a few schemes from the ones listed in Section 2.4 support it, with limited exposure and acceptance outside the geographical area of the issuing organizations.

3.2 QUALITY ASSURANCE

In today's software development landscape, organisations and individuals have difficulty in identifying the level of security of software products. Such information is important to gain trust but cannot be easily acknowledged from the outside. Some security functions (e.g. authentication methods) are visible for everybody, but that is only a really small part of the total set of security functionalities that a software may need to provide. For example, from the 286 controls of the OWASP Application Security Verification Standard Project [14], only 22 can be evaluated by a non-security expert. There seems to be information asymmetry between producers and consumers of software. Consumers cannot identify immediately the level of security of the software products they buy. Even worse, in commercial procurement procedures, (e.g. a mobile operator procures the development of a mobile application) organisations have difficulty firstly in setting the right requirements and secondly in evaluating the level of security of the software product they acquired.

The same information asymmetry causes assurance unclarity: it is often not clear to a non-expert what a certificate or an assessment report actually means with regards to the level of security or even the security risks mitigated for a certain software product. Secondly, because of the intricate supply chain we have today, assurance suffers from the complexity of the whole system. Software systems can consist of multiple subsystems, contain components from third parties, or even open source (that are in some cases publicly available). Lastly, the operational environment is also part of the security scope. That environment can be controlled by an external party if the software is (partly) provided as a service. These scope issues are not always apparent to non-experts, resulting in a mistaken perception on the scope of evaluated software components. For example, suppose a mobile app has been evaluated as compliant to a mobile application security standard, while at the same time the following parts were out of scope:

- The part of the application that runs on a webserver (which is often the case for apps).
- A third-party component because the supplier did not provide access to the source code.
- The operational environment in which this webserver is deployed (network, firewall, patching of the server etc.).

3.3 SUSTAINED TRUST ISSUES

Maintaining confidence in the security level of a software product over time is not directly addressed by existing assurance/certifications initiatives. Such challenges in sustained trust are there for three reasons:

- Software changes after an evaluation/certification may introduce new vulnerabilities (point in time issue).
- Even without software updates/changes, newly discovered vulnerabilities can be found at the software, or in its (open source/third-party) components/libraries.
- The software's operational environment in not covered directly by the same evaluation process and is also subject to changes/updates or newly discovered vulnerabilities.

Under some of the existing certification schemes, issued certificates become invalid after (significant) changes are performed to the evaluated product/software. A side effect is that, sometimes, vendors refrain from updating their software in order not to have to undergo an evaluation/certification process again, due to aforementioned updates. In such cases, the security certification can ironically be considered as making systems likely less secure.

3.4 ASSURANCE CLARITY BETWEEN PROCESS AND PRODUCT

Software development process assessment can provide a good indication on whether a software developing organization (or the corresponding supply chain) is able to develop and to maintain secure software, even though it has severe assurance limitations. This type of process assessment is often considered as a way to provide assurance that software remains secure, change upon change. This is an attractive idea, as it would reduce the effort in re-assessment or re-certification substantially. However, no matter how good a process is, or seems to be, mistakes can always be made. In other words: software process assessment is helpful but cannot provide the same level of assurance that products assessments can – unless the process features trustworthy product assessment of course.

Software product assessment, on the other side, provides insights only into the current level of security, with little assurance about the future. It is only a snapshot. This can be a reason for assessing the software development process as well. Let's look conceptually at what a development process entails. It is about having procedures, policies, culture and tools in place so that security is being built in from the start. It is not the total set of programming steps that developers take to write source code with security built in. If that would need to be assessed, then the best subject to assess would be the source code itself (the product). In other words: secure software process assessment is about investigating the developers' organisation.

Unfortunately, assessment of security in a software development process is hard to perform in a reliable way, as:

- A large part of the process effectiveness is determined and driven to a certain extent by the skills, and the knowledge of the people, and the actual priority that security gets in software development: aspects that are hard to measure and are not directly included in the popular secure SDLC standards. The resources dedicated to software development and quality are an additional driving factor.
- It is relatively easy to make a complex process like software development look good. There is a big difference between having a procedure and actually following it in practice and perform it well. For example: one can argue that peer review is being performed on source code, but how skilled are the reviewers, how much time do they have and is there a culture where honest feedback can be provided? These are aspects that are hard to measure and easy to claim. There is also a big difference between having a tool, using it, and using it well. Furthermore, there are vast differences between security tools when it comes to coverage and quality of results.

4. DRIVING SOFTWARE SECURITY IN THE EU

4.1 DEVELOP A COMMON REPOSITORY FOR SHARED SECURITY MEASURES

Aligning commonalities of requirements across different schemes prevents proliferation and fragmentation, while also making drafting and maintaining a scheme more efficient in terms of mitigating the risks. As such, the commonalities evolve in quality and efficiency of mitigating the risks over time. For such common ground in harmonization to be maintained and to be used, a mapping governance, documenting the overlaps, could greatly support scheme development and maintenance. Such a mapping could result in the definition of a common repository for shared security aspects (access control, authorization, encryption etc), threat models and approaches against known adversary tactics, over different schemes as part of obligations introduced by Cybersecurity Act Article 55 on Supplementary cybersecurity information. Such a repository could be extended to become a knowledge sharing platform which would greatly help to reach the mentioned goals. Next to the technical requirements, this repository could also keep metadata such as: mapping to existing standards, revision history, related threats, and implementation guides for different technologies, or references to those.

Manufacturer(s) or provider(s) of certified ICT products, ICT services or ICT processes, should consider the deployment and maintenance of repositories not only for publicly disclosed vulnerabilities but also for shared security aspects of certified products, services and processes towards aligning on requirement commonalities and ways to mitigate common security risks.

4.2 IMPROVE TECHNICAL STANDARDS LANDSCAPE

The Cybersecurity Act reinforces the need for referencing international, European or national standards within the EU cybersecurity certifications schemes. Picking the best source requires taking into account many factors: comprehensiveness, clarity and fitness to the scope of the cybersecurity certification scheme. Despite the ongoing standardization efforts, gaps in standardization towards the Cybersecurity Act goals still exist [13] [15]. Such gaps present risks that additional standardization efforts may mitigate, but overlaps in standardization efforts present risks that may be mitigated only with a coordinated approach. A good starting point could be the Union Rolling Work Programme (CSA Art. 47), the first edition of which is expected to be published by mid-2020.

Following the publication of URWP, SDOs and ESOs should coordinate on the priority areas they can support, put forward standardization activities to benefit the future developed schemes and communicate periodically such planning to the EC and relevant CSA stakeholders.

4.3 PROVIDE ASSURANCE IN THE ENGINEERING PROCESS

When analysing available SDLC (Software Development Lifecycle) standards it becomes clear that there is consensus on what activities should be included. This consensus centers on the key activities that can be distinguished in the development lifecycle like: training, requirements, coding guidelines, design, design review, threat modelling, secure verification (automated testing, static analysis tools, test tools, manual code review and penetration testing), dependency management, incident management, vulnerability management and environment hardening.

Many of the existing process standards include maturity levels, supporting organizations to assess their own maturity and to get guidance on improving that maturity, and also appreciating that there is no line to draw when it comes to engineering. As described earlier, process assurance provides limited certainty on the security of the software that comes out of the process. One way to address this is to include provisions, depending also on the level of assurance and the scope of the scheme, on the verification of the software and the process used. Such an assurance on the process would not only improve the end product but could also benefit the whole supply chain of the aforementioned software product or the supply chain that the software product is involved. Lastly, assurance on the process could also be considered as a weighting factor, similar to certification, with regards to mitigating (possible) liabilities or as a criterion on selection of supplier(s).

EU cybersecurity certification schemes for products, services and processes should include, to the extent possible, not only requirements for the end product/service/process but also assurance for the engineering process, by setting process guidelines for software development, maintenance and operation.

4.4 INCREASED SCHEMES APPLICABILITY AND CLEARLY COMMUNICATED ASSUMPTIONS

In order to constitute the evaluation and (eventually) certification processes appealing to software developers, it is important to provide assurance levels that are appropriate for the different levels of risk given the context and the expected domain of use. Since the EU cybersecurity certification framework envisions three assurance levels, one of which allows for self-conformity assessment, it seems beneficial to retain the notion of lightweight conformity assessment.. By providing lightweight approaches, the cost of re-certification (through third parties) will no longer prevent companies to release updated software. Further to that, mechanisms can be put into place to base the conformity assessment on a change analysis of the software, similar to existing national approaches. Additionally, certification of the secure software development process is also a way to provide more sustained trust. Such a certificate delivers meta-assurance that sets requirements on the level of assurance taking place in the development process.

Given the magnitude of the context and the domain(s) in which software products can be used, the aforementioned certification schemes should be applicable horizontally, to the widest extent possible. In cases where this is not feasible, they could be considered as intuitive guidelines and best practices by providing a clear indication of the risks they mitigate and the assumptions made within the scope of the scheme.

During the development of EU cybersecurity certification schemes, lightweight conformity assessment for assurance level basic should be considered as a partial response to the existing fragmented landscape of software development and maintenance.

Software and product manufacturers should put forward their experience and expertise and promote the uptake of EU cybersecurity certification schemes, including the self-assessment components.

During the identification of strategic priority areas, under the URWP, and the development of the EU cybersecurity certification schemes, EC, ENISA, SCCG and ECCG should ensure that assumptions on the scope, application area, mitigated threats and achieved security characteristics achieved are clearly communicated to the end users of the framework.

5. REFERENCES

1. Verizon: 2019 Data Breach Investigations Report. (2019)

2. THALES: 2019 Thales Data Threat Report – Global Edition. (2019)

3. ETSI: ETSI TS 103 645 V1.1.1 (2019-02)- Cyber Security for Consumer Internet of Things. (2019)

4. Security Boulevard Available at: https://securityboulevard.com/2019/10/60-of-breaches-in-2019-involved-unpatched-vulnerabilities/

5. European Union: Regulation (EU) 2019/881 of the European Parliament and of the Council of 17 April 2019 on ENISA and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013 (Cybersecurity Act). (2019)

6. Australian Cyber Security Centre: Guidelines for Software Development. (2019)

7. ENISA: Good Practices for Security of IoT - Secure Software Development Lifecycle. (2019)

8. SafeCode: Fundamental Practices for Secure Software Development. (2018)

9. ISO/IEC: ISO/IEC 27000:2018 Information technology — Security techniques — Information security management systems — Overview and vocabulary. (2018)

10. Pigoski, T.: Software Maintenance. (2001)

11. ISO/IEC: ISO/IEC 27034-1:2011 Information technology — Security techniques — Application security — Part 1: Overview and concepts. (2011)

12. Shiklo, B.: 8 Software Development Models: Sliced, Diced and Organized in Charts. (Accessed November 2019) Available at: https://www.scnsoft.com/blog/software-development-models

13. ENISA: ICT security certification opportunities in the healthcare sector. (2019)

14. OWASP: Application Security Verification Standard 4.0. (2019)

15. ENISA: Guidance and gaps analysis for European standardisation. (2019)

16. PCI Security Standards Council: PCI Mobile Payment Acceptance Security Guidelines for Merchants as End-Users. (2014)

TP-03-20-206-EN-N

ABOUT ENISA

The mission of the European Union Agency for Cybersecurity (ENISA) is to achieve a high common level of cybersecurity across the Union, by actively supporting Member States, Union institutions, bodies, offices and agencies in improving cybersecurity. We contribute to policy development and implementation, support capacity building and preparedness, facilitate operational cooperation at Union level, enhance the trustworthiness of ICT products, services and processes by rolling out cybersecurity certification schemes, enable knowledge sharing, research, innovation and awareness building, whilst developing cross-border communities. Our goal is to strengthen trust in the connected economy, boost resilience of the Union's infrastructure and services and keep our society cyber secure. More information about ENISA and its work can be found at www.enisa.europa.eu.

ENISA
European Union Agency for Cybersecurity

Athens Office
1 Vasilissis Sofias Str
151 24 Marousi, Attiki, Greece

Heraklion office
95 Nikolaou Plastira
700 13 Vassilika Vouton, Heraklion, Greece

enisa.europa.eu

ISBN: 978-92-9204-343-8
DOI: 10.2824/81950