D4.2

VESSEDIA approach for security evaluation

Project number:	731453
Project acronym:	VESSEDIA
Project title:	Verification engineering of safety and security critical dynamic industrial applications
Start date of the project:	1st January, 2017
Duration:	36 months
Programme:	H2020-DS-2016-2017

Deliverable type:	Report
Deliverable reference number:	DS-01-731453 / D4.2 / 3.0
Work package contributing to the deliverable:	WP4
Due date:	June 2018 – M18
Actual submission date:	16 October 2018

Responsible organisation:	SLAB
Editor:	Balázs Berkes
Dissemination level:	PU
Revision:	V3.0

Abstract:	This document describes a proposed security evaluation methodology for VESSEDIA project, as declared in the work of Task 4.2 “Security Evaluation” inside WP4. The methodology defined here is to be executed in later phases of the project and will be documented in following deliverables. This document also collects and describes the state-of-the-art of certification frameworks, and the VESSEDIA and STANCE tools developed by partners that will be used in the security evaluation in later phases.
Keywords:	Security, evaluation, IoT, certification, verification, tools, tooling

The project VESSEDIA has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731453.

Editor

Balázs BERKES (SLAB)

Contributors (ordered according to beneficiary numbers)

Mounir KELLIL (CEA)

Virgile PREVOSTO (CEA)

Dillon PARIENTE (DA)

Gergely EBERHARDT (SLAB)

Vendel LÁSZLÓ (SLAB)

Balázs BERKES (SLAB)

Nicolas ZILIO (AMO)

Executive Summary

The current deliverable describes the work carried out towards the Objectives for WP4 set forth in DoA, namely to review existing security evaluation methodologies in the relevant domains, and to investigate how they can benefit from the integration of methods and tools developed during the VESSEDIA project. Relationship to the Common Criteria security standard (CC) is also discussed.

The document describes a proposed security evaluation methodology for VESSEDIA project, as declared in the work of Task 4.2 inside WP4. The methodology defined here will be executed in later phases of the project and will be documented in following deliverables.

This document also collects and describes the VESSEDIA and STANCE tools developed by partners that will be used in the security evaluation in later phases. The execution of the methodology will also provide results regarding the use of VESSEDIA tools in the security evaluations, contributing to further deliverables.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 VESSEDIA motivation and background

The VESSEDIA project aims to bring safety and security to the next generation of software applications and Internet connected devices. In our rapidly changing world, the Internet has been the source of many benefits for individuals and companies alike, transforming entire industries. With this new technology, capable of connecting billions of devices and people together, new threats have also appeared – threats VESSEDIA will help software developers address in order to create connected applications that are safe and secure. VESSEDIA proposes to enhance and scale up modern software analysis tools, in particular the mostly used open-source Frama-C analysis platform, to make them useful and accessible to a wider audience of developers of connected applications. At the forefront of connected applications is the Internet of Things (or IoT for short), which has undergone explosive growth and where security risks have become all too real. VESSEDIA will focus on this domain to demonstrate the benefits our tools bring to the table when developing connected applications. VESSEDIA will tackle this challenge by 1) developing a methodology that makes it possible to adopt and use source code analysis tools as efficiently and with similar benefits as it is already possible in the case of highly-critical applications, 2) enhancing the Frama-C toolbox to enable efficient and fast implementation, 3) demonstrating the capabilities of the new toolbox on typical IoT applications, including an IoT Operating System (Contiki), 4) developing a standardisation plan for generalising the use of the toolbox, 5) contributing to the Common Criteria certification process, and 6) defining a “Verified in Europe” label for validating software products with European technologies such as Frama-C.

1.2 Role of the deliverable

This document analyses current evaluation and certification frameworks and existing security evaluation methodologies and describes a proposed security evaluation methodology for VESSEDIA project, as declared in the work of Task 4.2 inside WP4. The methodology defined here will be executed in later phases of the project and will be documented in the following deliverables.

This document also collects and describes the VESSEDIA and STANCE tools developed by partners that will be used in the security evaluation in later phases. The execution of the methodology will also provide results regarding the use of VESSEDIA tools in the security evaluations, contributing to further deliverables.

1.3 Structure of the document

The document can be divided into 4 major parts:

Chapter 2 describes the State-of-the-Art regarding certification approaches useful for IoT devices, as well as defines the Security Certification Level (SCL) to be applied in VESSEDIA security evaluations.

In Chapter 3 we specify the security evaluation methodology to be applied in later phases and the outcomes expected in later deliverables.

VESSEDIA tools that will be developed by project partners and also used in the security evaluations are enumerated in Chapter 4.

This project was preceded by the STANCE project where similar tools were developed. Chapter 5 lists the tools that are considered relevant to the VESSEDIA work ongoing.

1.4 Related deliverables

In VESSEDIA, a set of VESSEDIA tools and corresponding Metrics are developed – see also D4.1 [3] for Metrics.

Parallel to the execution of the evaluation plan according to the methodology described in this document, the automated testing tools and the quality of the Metrics generated will also be assessed throughout the work of this work package. D4.3 [4], D4.4 [5], and D4.5 [6] will describe different aspects of the use of VESSEDIA tools.

The Evaluation itself will be carried out on selected WP5 Use case components at the final stage of the project and results will be documented in the final deliverable of WP4: D4.6 [7].

Chapter 2 Certifications for IoT devices

Chapter 2 describes the State-of-the-Art regarding certification approaches useful for IoT devices, as well as defines the Security Certification Level (SCL) to be applied in VESSEDIA security evaluations.

2.1 Common Criteria for IoT devices

Common Criteria (CC) is an international standard used in computer security for claiming and validating the security level of a product. It provides assurance that the process of specification, implementation and evaluation of a computer security product has been conducted in a rigorous, standard and repeatable manner.

CC defines seven Evaluation Assurance Levels:

- **EAL1**: Functionally Tested;
- **EAL2**: Structurally Tested;
- **EAL3**: Methodically Tested and Checked;
- **EAL4**: Methodically Designed, Tested and Reviewed;
- **EAL5**: Semiformally Designed and Tested;
- **EAL6**: Semiformally Verified Design and Tested;
- **EAL7**: Formally Verified Design and Tested.

Obviously, as the assurance level increases, the evaluation tasks increase, along with the evaluation price and duration.

An evaluation is linked to a Security Target, describing the Target of Evaluation (TOE). The TOE is the product being evaluated. The TOE perimeter, in its logical and physical boundaries is clearly defined with many details. The evaluation tasks also depend on the TOE perimeter.

With those two main elements: assurance level and ToE perimeter, a CC evaluation can last from several months to one year or more. Table 1 presents the costs and efforts required for each level of certification. For instance, an EAL3 evaluation costs between 57 000 and 151 000 €, and the whole process of evaluation and certification lasts between 6 and 13 months.

	Effort (man-days)		Cost		Duration of the process of Evaluation and Certification (in months)	
Level	Min	Max	Min	Max	Min	Max
EAL1	33	72	27 000 €	74 000 €	4	9
EAL2	62	129	50 000 €	131 000 €	5	12
EAL3	70	148	57 000 €	151 000 €	6	13
EAL4	101	206	81 000 €	209 000 €	7	18
EAL5	121	238	97 000 €	241 000 €	9	23

Table 1: Estimation of Costs and Efforts for Common Criteria from EAL1 to EAL5

As we can see in the table above, obtaining a Common Criteria certification may last for several months and cost tens of thousands of Euros. Many IoT products have quite simple functionalities.

Most of them only have authentication functionality for administration tasks and their communications protected by TLS. These security functions are enough to protect most of IoT products. Since then, the question may be asked if Common Criteria process is relevant and well adapted to treat with this kind of product.

On one hand, some of the latest researches in applying Common Criteria for IoT yielded interesting results. Kang and Kim (2017) [8] shared the case of EAL2 CC certification through structural testing for security and reliability on a specific Smart TV.

On the other hand, in an IoT system, it may be difficult to talk about a product per se. IoT is best seen as an interaction of many devices. One possibility to define the TOE perimeter would be to include many devices, and the interactions between them. That would be very costly to evaluate. Moreover, the different devices might come from different developers, and the collaborative task to write a common Security Target would be very tedious.

Also, a new device might be added to the system in the future. The certified IoT system without this new device would become obsolete. Then, a different possibility for the ToE perimeter would be to evaluate each product of the IoT system independently. Since a CC lasts at least several months, the evaluation cost for IoT system comprising several devices would be too high.

The EU-funded ARMOUR Project[1] delivers a broad analysis on the issues and challenges of security certification schemes, with some especially true in the Internet of Things environment. It includes guidelines to overcome those issues. The VESSEDIA methodology can provide value in consideration of the tools used in the safety and security verification process.

To conclude, Common Criteria in its current form, is not very suitable for IoT systems.

2.2 Verified in Europe

The VESSEDIA project aims at initiating a “Verified in Europe” label. This label will contribute to make safety and security tools (i.e. Frama-C) as widely recognized by the developer and security communities. The label will be discussed, defined and designed as a new standard to be supported by the members of the VESSEDIA Advisory Board. The label is benchmarked on the candidate ISO Standard that is developed within VESSEDIA in WP6: D6.4.

The VESSEDIA candidate ISO standard itself defines capabilities of and requirements for software safety and security verification tools. In this regard, software that fulfils the requirements will obtain the “Verified in Europe” label. Details and modalities of the certification process are not set yet, at the time of delivery of D4.2, but will be done within WP6: D6.4 (M36).

Figure 1 illustrates where the “Verified in Europe” label improves end-users’ (client/sponsor) visibility of the product quality and quality-in-use of the labelled system/software as compared with a non-labelled. The label itself reflects the efforts put in the security and safety verification process, and more precisely, which tool capabilities have supported it.

Generally, the label and its requirements for certification shall improve safety and security in industrial applications, support the competitiveness of software/systems, and improve end users’ trust in such applications.

Figure 2 shows the design of the label.

[1] ARMOUR – Large-Scale Experiments of IoT Security Trust, www.armour-project.eu, Project Reference 688237

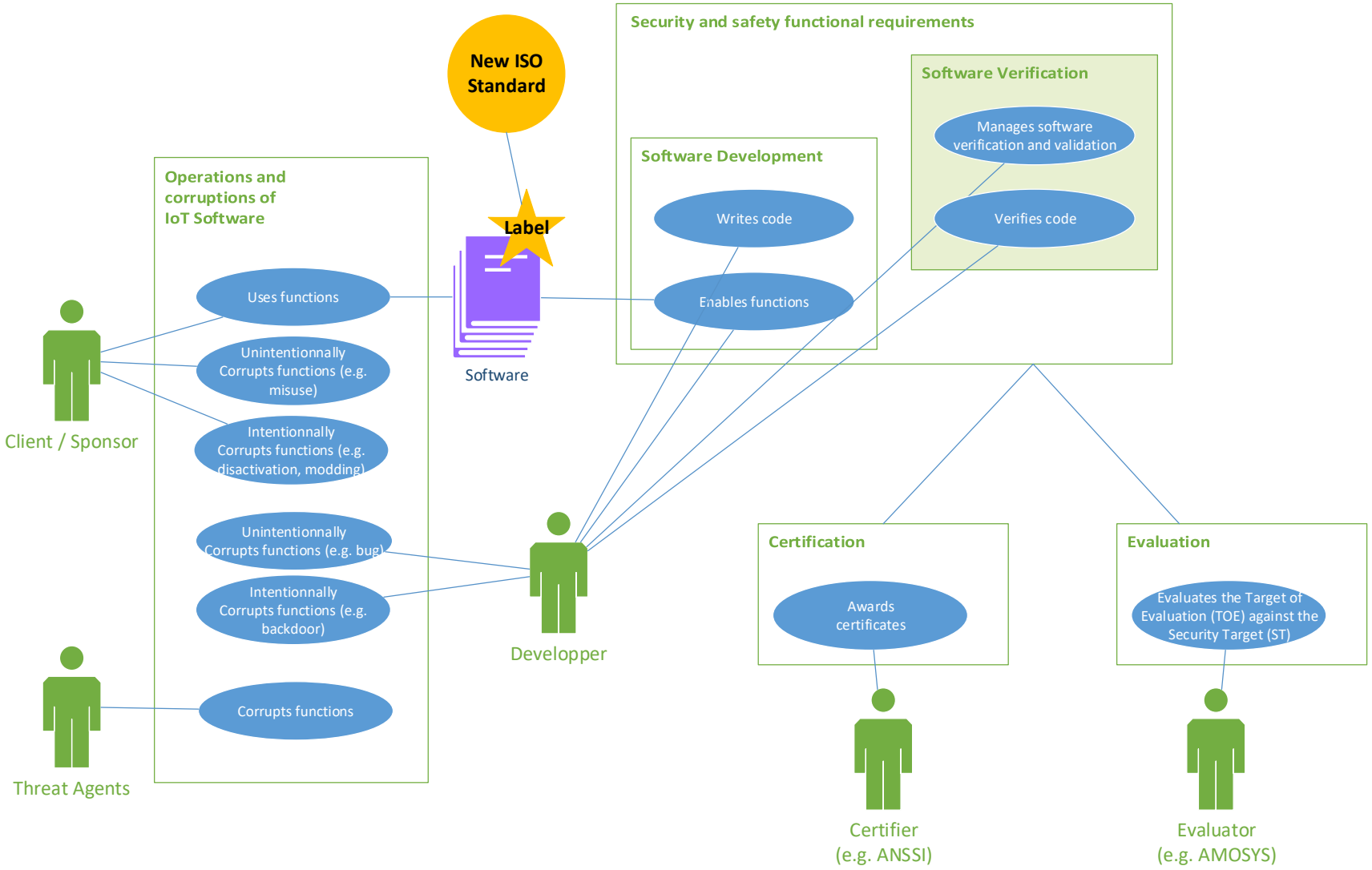

Figure 1: the role of the Verified in Europe label

Figure 2: the design of Verified in Europe label

2.3 Evaluation and Certification frameworks

2.3.1 European Cybersecurity Certification Framework and European Trust Label (ENISA)

The EC established and supports European Network and Information Security Agency (ENISA), according to [10], in order to, amongst others:

> *"(13) The Agency should carry out the tasks conferred on it by legal acts of the Union in the field of electronic communications and, in general, contribute to an enhanced level of security of electronic communications as well as of privacy and personal data protection by, among other things, providing expertise and advice, and promoting the exchange of best practices, and offering policy suggestions."*

2.3.1.1 European Trust Label for connected devices

ENISA published its position on the necessity of a European Trust Label, together with industrial partners Infineon, NXP, and STMicroelectronics - see Common Position On Cybersecurity (Dec. 2016) [11].

The report recommends the following:

> *"The European Commission should define a policy framework for ensuring minimal security requirements for connected devices. The development of European security standards needs to become more efficient and/or adapted to new circumstances related to IoT. Based on those requirements a European scheme for certification and the development of an associated trust label should be evaluated."*

Further relevant details regarding the challenges and recommendations about standardization and certification are as follows:

- *"A currently well-established certification of security products is **Common Criteria** (CC, ISO/IEC 15408). It has been implemented in many EU Member States and other countries and benefits from a dedicated mutual recognition agreement (SOG-IS MRA) between 13 EU member states. This should be extended ideally to all Member States. Also SOG-IS MRA covers high assurance security levels, a key asset on which the EU should build. The MRA and the way CC is applied require adjustments in order to stay attractive for the development of modern applications and agile development methodologies. In addition, for customers/applications not requiring full CC certification or those willing to accept lower security levels while maintaining the vulnerability assessment of the security solutions, **a baseline security certification** (essentially a "lightweight" certification) for ICT products should be developed and agreed by all MSs, addressing IoT, Commercial of-the-shelf (COTS) and products with short life cycle. Certification should use standardized security requirements for connected devices as reference."*
- *"The introduction of a **European trust label for connected devices** should be built on defined baseline security requirements and existing internationally recognized certification schemes. Such a European trust label should make the underlying certification transparent, also including the targeted security need. This should be supported by a **European Certification Framework**."*

2.3.1.2 European Cybersecurity Certification Framework

The most current policy making act forming the Digital Single Market in the European Commission is the formation of the EU cybersecurity certification framework [12].

On 13 September 2017 the Commission issued a proposal for a regulation on ENISA, the "EU Cybersecurity Agency", and on Information and Communication Technology cybersecurity certification ("Cybersecurity Act") [13].

According to the policy description:

> *The proposed certification framework will provide EU-wide certification schemes as a comprehensive set of rules, technical requirements, standards and procedures. This will be based on agreement at EU level for the evaluation of the security properties of a specific ICT-based product or service e.g. smart cards.*
>
> *The certification will attest that ICT products and services that have been certified in accordance with such a scheme comply with specified cybersecurity requirements. The resulting certificate will be recognized in all Member States, making it easier for businesses to trade across borders and for purchasers to understand the security features of the product or service.*
>
> *The schemes proposed in the future European framework will rely as much as possible on international standards as a way to avoid creating trade barriers and ensuring coherence with international initiatives.*

According to Article 46 Assurance levels of European cybersecurity certification schemes in the "Cybersecurity Act" proposal [13]:

1. *A European cybersecurity certification scheme may specify one or more of the following assurance levels: basic, substantial and/or high, for ICT products and services issued under that scheme.*
2. *The assurance levels basic, substantial and high shall meet the following criteria respectively:*

a. *assurance level **basic** shall refer to a certificate issued in the context of a European cybersecurity certification scheme, which provides a limited degree of confidence in the claimed or asserted cybersecurity qualities of an ICT product or service, and is characterised with reference to technical specifications, standards and procedures related thereto, including technical controls, the purpose of which is to decrease the risk of cybersecurity incidents;*

b. *assurance level **substantial** shall refer to a certificate issued in the context of a European cybersecurity certification scheme, which provides a substantial degree of confidence in the claimed or asserted cybersecurity qualities of an ICT product or service, and is characterised with reference to technical specifications, standards and procedures related thereto, including technical controls, the purpose of which is to decrease substantially the risk of cybersecurity incidents;*

c. *assurance level **high** shall refer to a certificate issued in the context of a European cybersecurity certification scheme, which provides a higher degree of confidence in the claimed or asserted cybersecurity qualities of an ICT product or service than certificates with the assurance level substantial, and is characterised with reference to technical specifications, standards and procedures related thereto, including technical controls, the purpose of which is to prevent cybersecurity incidents.*

2.3.1.3 Baseline Security Recommendations for IoT

"ENISA Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures" was issued in November 2017 [20].

This document researches and describes the following:

- The IoT paradigm
  - Elements of IoT
  - Security considerations
  - Challenge of defining horizontal baseline security measures
  - Architecture
  - Asset taxonomy
- Threats and risk analysis
  - Security incidents
  - Threat taxonomy
  - Examples of IoT cyber security attack scenarios
  - Critical attack scenarios
- Security measures and good practices
  - Policies
  - Organisational, People and Process measures
  - Technical Measures
- Gaps analysis
- High-level recommendations to improve IoT cybersecurity
- Annex A: Detailed Security measures / Good practices
- Annex B: Security measures and threats mapping
- Annex C: Security standards and references reviewed
- Annex D: Description of indicative IoT security incidents

From this study, the Asset taxonomy provides a practical collection of functionalities that will be taken into account in defining the Applicability field for the evaluation activities – see chapter 3.1. Please note slight difference in the terminology – we consider the elements described in the Asset taxonomy in [20] as functionalities, while we use the term 'valuable assets' for entities within the system that need to be protected.

Threat taxonomy and impact (and affected assets) will also be taken into account in our own risk analysis and threat modelling (see 3.6.2 and 3.3, respectively). The security evaluation will be carried

out, and will be described in later deliverables (i.e., [7]). The results are expected to provide valuable feedback with regard to the applicability of the actual threat model.

2.3.2 European Cyber Security Certification (ECSO)

The European Cyber Security Organization (ECSO) published its Meta-Scheme Approach in December 2017 - see [21].

The document studies the State-of-the-Art and challenges with regard to cyber-security certifications and defines objectives with regard to a future-proof certification methodology.

The meta-scheme proposed in the document specifies five levels of assurance in two groups as follows:

- Base Group: sector-agnostic standardized base layer evaluation, providing two levels of assurance:
  - Level E (Entry): based on self-assessment (done by the vendor) where the minimum scope of security functionality to be fulfilled is basically sector-agnostic and can be seen more as a black-box approach.
  - Level D (Basic): is the same as Entry but with the difference that an accredited third party did the assessment.
- Advanced Group: specifies more advanced assessment by an accredited third party where the minimum scope of security functionality is sector specific:
  - Level C (Enhanced Basic): assessment depth is starting going beyond a black-box view and the scope of security functionality is sector specific; the assessment requires resistance against an enhanced basic attack potential.
  - Level B (Moderate): assessment depth can be considered as a “grey”-box7 approach and the scope of security functionality is getting clearly higher than for Enhanced; the assessment requires resistance against a moderate attack potential.
  - Level A (High): assessment depth is a white-box approach and the scope of security functionality is getting clearly very high; the assessment requires resistance against high attack potential.

The above categories define different approaches and activities for different levels. Comparing to the VESSEDIA approach as described in this document, the Level D (Basic) assessment referred above is similar to the certification process for Security Certification Levels (SCL) for lower critical devices described in 2.4.

The document further describes the creation of a “Generalized Protection Profile” and a related “Generalized Security Target”, which cover:

Generalized Protection Profile (GPP):

- Security Problem Definition (Threats, Assets, Assumptions, Policies) from risk/threat assessment
- Security Objectives (derived from the Security Problem Definition)
- Security Services and Features derived from the objectives
- Instantiation of the levels
  - Selection of the schemes
  - Visual representation of the minimum required scope of security functionality per level w.r.t. Security Services & Features (e.g. radar diagram).
  - Additional evaluation steps required.

Generalized Security Target (GST):

- Reference to the GPP which is used as a basis for evaluation and definition/selection of parts which were left open by the GPP (i.e. some parts require instantiation).
- Additional claims which are specific to the item under evaluation (this the vendor might want to use to differentiate from other vendors).
- Key security features and services fulfilling the claims (this might include additional ones compared to the GPP).

- Visual representation of the claims (GPP + GST in combination).

The above terms closely follow nomenclature from Common Criteria, and are similar in role as Protection Profile and Security Target, respectively (see also 2.1).

Finally, the document describes the **European Cyber Security Certificate (ECSC)**, which is analogous to the *Verified in Europe* label defined in the VESSEDIA Project (see also 2.2).

The ECSC containing the certification and report would have the following aspects:

- **Label** including e.g. a QR-code or NFC-tag for navigation to an (online) version of the ECSC.
- **Main attributes of the evaluation** such as the exact name of the product, sector identifier, unique identification of GPP, GST, the Accredited Third Party used, validity of certificate, list of guidance documentation evaluated, evaluated configuration options, etc.
- **Scope of Security Functionality** and respective evaluation results in simple visual form (e.g. radar diagram).
- **List of subsequent certificates** used (be it ECSC or others).

2.3.3 IoT Security Compliance Framework (IoTSF)

IoT Security Foundation (IoTSF), released the 1.1 version of the IoT Security Compliance Framework documents in Dec. 2017 [14].

This framework describes best practices and security guidance to organizations that want to build secure IoT products.

The document is a checklist to guide an organization through the assurance process and gather structured evidence to demonstrate conformance with best practice. Additional Best Practice Guidelines are provided by the Foundation to help better understanding. Further background information is contained in linked reference documents on the IoTSF website.

The working groups of the IoTSF foundation continue work on the compliance framework, and at the time of writing this report, the 2.x Draft version is available to members, together with a working version of the IoTSF Trustmark Specification Release 1, which will, similarly to the Verified in Europe label (see 2.2), provide a mark that would help the end-users to recognize efforts carried out in the context of the compliance framework.

2.3.4 CSPN and Security Visa (ANSSI)

2.3.4.1 First Level Security Certification (CSPN - Certification de Sécurité de Premier Niveau)

The French National Cybersecurity Agency (Agence nationale de la sécurité des systèmes d'information – ANSSI) defined FIRST LEVEL SECURITY CERTIFICATION (CSPN - CERTIFICATION DE SÉCURITÉ DE PREMIER NIVEAU) [21] as a lightweight alternative to Common Criteria. As of March 2018, this scheme has been used to evaluate 254 products since 2008, from which 116 were certified.

The procedure is detailed in [23], it is largely based on ANSSI's certification body and licensed evaluation facilities. CSPN is claimed to provide evaluation methodology to check "elementary" intrinsic resistance against security threats, but does not claim guarantees that there will be no exploitable vulnerabilities in the product. As also described in relation to the Security VISA (see 2.3.4.2), CSPN compliments Common Criteria to provide lower-level assurance.

As stated in [24], the following two objectives are targeted during a CSPN evaluation:

- To check that the product conforms to its security specifications;
- To determine the effectiveness of the security functionalities proposed by the product.

The following content is required for a security target (specified in the Evaluation Technical Report; ETR):

- A clear identification of the product to be evaluated;
- A rationale for the product, describing in natural language the use for which the product has been designed, by whom and in which usage context it is supposed to be used;
- The technical environment in which the product operates (computer type, operating system, etc.);
- The sensitive assets that the product must protect;
- The environmental measures;
- The threats against which the product offers protection;
- The security functions implemented by the product to counter the identified threats. These are the functions which will be the subject of the evaluation.

Evaluation Criteria is divided into the following phases, and Evaluator's tasks are defined for each phase:

- Phase 1 - security target analysis
- Phase 2 – product installation
- Phase 3 – conformity analysis – documentation analysis
- Phase 4 – conformity analysis – source code review (if available)
- Phase 5 – conformity analysis – product testing
- Phase 6 - resistance of the mechanisms/functions
- Phase 7 – vulnerability analysis (intrinsic, construction, exploitation, etc.)
- Phase 7a – host system vulnerability analysis
- Phase 8 – ease of use analysis
- Phase 9 – meetings with the developers
- Phase 10 – cryptography evaluation (if the product implements cryptographic mechanisms)

Table 2 presents the costs and efforts required for each level of certification.

	Effort (man-days)		Cost		Duration of the process of Evaluation and Certification (in months)	
Level	Min	Max	Min	Max	Min	Max
CSPN	29	43	23 000 €	43 000 €	3	7

Table 2: Costs and Efforts for CSPN evaluations

2.3.4.2 Security VISA

The French information security agency (ANSSI) developed the CSPN certification as a lightweight alternative to CC certifications. In January 2018, ANSSI also announced Security VISA [25] to certify and/or qualify products according to French and/or European criteria.

The ANSSI security visas meet three objectives:

- Regulatory objectives: to meet the French or European regulations that enforce the use of solutions guaranteeing a tried and tested level of robustness.
- Contractual objectives: to provide a solution for public or private contractors who demand that the solutions they use have obtained an ANSSI security Visa.
- Commercial objectives: to enable a product supplier or service provider, as well as the final users of these solutions to stand out from the competition by guaranteeing a certain level of robustness.

Certification is defined as:

> *" Certification demonstrates the robustness of products that have been evaluated through penetration tests by a third-party evaluator under the authority of ANSSI, according to a process and framework in order to provide the best level of security given the market and technological requirements."*

The French system provides two types of certification. Each of them comes with specific methods and criteria:

- Common Criteria (CC) certification

First Level Security Certification (CSPN - Certification de Sécurité de Premier Niveau) Qualification, in turn, is defined as:

> *"Attesting compliance with the regulatory, technical and security requirements promoted by ANSSI, the qualification is the French state's recommendation of cyber security products or services that have been proven and approved by ANSSI.*
>
> *The qualification demonstrates a product's robustness, the service provider's competency, and guarantees the product or service supplier's commitment to comply with trust criteria.*
>
> *The qualification aims at providing government authorities and operators of vital importance with the products and services that meet their information systems security requirements."*

For products, there are three levels of qualification. Each of them corresponds to a level of security and compliance with trust criteria: basic, standard, and enhanced. For services, there are several families of services defined in the regulatory framework: audit, incident response, incident detection, cloud computing, and trusted digital services (electronic certification, electronic time stamping, approval and conservation of electronic signatures and seals, and electronic registered delivery services).

2.4 Security Certification Levels for lower critical devices

In accordance with the current emerging policies briefly described in the previous chapter 2.3, most certification frameworks prescribe different verification levels for different security targets.

Cost/efforts spans a wide range and should be considered in planning necessary activities for each target, as the estimated time and effort in separate frameworks can be compared between tables Table 1 and Table 2.

In our document, we define Security Certification Levels (see 2.5) to cover the range from the lowest end where (non-formal) evaluation is meaningful up to the level where formal verification according to other schemes like CC is necessary.

In VESSEDIA use cases, we specify the class of lower critical devices, and will describe the scope of evaluation for a related Applicability Field – see 3.1 for general description. A hypothetical "Internet Connected Box" (ICB) device will be defined as a – in some sense, generic – representation of the class of lower critical devices.

Typical assets were collected for IoT devices in [1], and for each device class, can be considered if present. The following security objectives were found for common IoT assets:

Using the CIA triad, the following security objectives can be defined for the collected assets	Confidentiality	Integrity	Availability
Personally identifiable information	X		
Credentials	X	X	X
Device settings	X	X	X
Application data	X	X	X
Device identification data	X	X	X
Sensor data	X	X	
Firmware		X	X
Server software		X	X
Mobile application		X	X
JTAG key	X		
Fused secrets	X		
Private keys	X	X	
Certificates		X	
Encryption keys	X		

Table 3: Security objectives for common IoT assets (from D1.1 pp15, [1])

In further work in WP4, we will carry out the security evaluation according to Chapter 3 on suitable components selected from the WP5 use cases. Definition of the Applicability Field (see 3.1) for lower critical devices will be carried out, comprising the threat modelling and risk analysis built on the assets, security objectives, threats, and risks collected, as well as the Test Case Specification, that will provide test methodology and evaluation plan.

2.4.1 Evaluation for SCL Certification

One of the expected valuable outcomes of the evaluation will be the SCL Certification level associated with the Target. See calculation at Section 2.5.2 and projected estimated attacker costs at 2.5.1. Although the SCL levels are specified up to SCL 10 corresponding to 50 days of estimated attack effort, actual evaluation effort will limit the level attainable in practice, and it is expected that the feasibility of the evaluation in VESSEDIA will set a hard limit on the levels targeted.

In commercial evaluation activities, we observed that generic lower-critical internet connected devices (such as routers, security cameras, etc.) can be reasonably (and feasibly) evaluated up to level SCL 6 approximately, with a few notable exceptions where some valuable assets need to be protected by the device to a higher level. In such cases, hybrid evaluation and certification efforts were required including systematic and more formal evaluation, targeting both the design and the implementation of security mechanisms protecting the valuable assets.

Thus, we advise to apply this certification scheme in VESSEDIA only for the lower- to medium-critical devices. Higher-security devices will need to be reviewed according to more formal methodologies, for which the project studies application of the Common Criteria (see 2.1 and 2.5.4). The work package WP4 will provide a further, more detailed study on the application of VESSEDIA tools with Common Criteria in D4.4 [5].

2.5 Defining Security Certification Level (SCL)

2.5.1 Definition of certification levels

SEARCH-LAB defined Security Certification Levels for its commercial evaluation work results. The SCL value has the following properties:

- Proportional to expected time frame, expert days, and costs for potential attackers mounting an attack against the valuable assets of the device
  - E.g. SCL 4: approximately 5 days of work by an expert attacker, ~10,000 USD total cost
- Can be scaled throughout a wide range of the evaluation efforts, proportionally to the required assurance level
  - Up to SCL 5: e.g., 10 Expert Days of evaluation work, within 3 weeks of time, with two physical samples, in a black-box manner

In the following, we summarize our past experience regarding the relation between SCL values defined, related Expert Days of evaluation, requirements for evaluation, and attacker efforts (estimated as Cost and Expert Days of an attacker).

We have devised an experimental calculation for approximating the equivalent SCL value (estimated protection level against theoretical attacks) based on the evaluation effort and the number of findings for different levels that resulted from systematic search by the evaluator. The calculation is based on past evaluation experience, where we were not only able to evaluate the targets, but also to collect past evidence on vulnerabilities and their estimated attacker effort and cost.

The following table - Table 4: Security Certification Level values - shows the relation defined between the SCL level and a theoretical attacker's effort and costs. The actual day and cost values included in this table are generalized estimations only, based on an evaluation sample count that gets statistically less relevant above SCL level 4.

Security Certification Level	Attack Time Frame	Expert Days of an Attacker	Additional Costs of an Attacker	Total Cost of Successful Attack
SCL 0 (FAILED)		< 1 days	< 100 EUR	< 1 000 EUR
SCL 1	one day	1 day	500 EUR	1 000 EUR
SCL 2		2 days	1 000 EUR	3 000 EUR
SCL 3		3 days	2 000 EUR	5 000 EUR
SCL 4	one week	5 days	5 000 EUR	10 000 EUR
SCL 5		7 days	8 000 EUR	15 000 EUR
SCL 6	two weeks	10 days	10 000 EUR	20 000 EUR
SCL 7		15 days	20 000 EUR	35 000 EUR
SCL 8	one month	20 days	25 000 EUR	50 000 EUR
SCL 9		50 days	50 000 EUR	100 000 EUR
SCL 10	> two months	>50 days	> 50 000 EUR	> 100 000 EUR

Table 4: Security Certification Level values

In the following, we show an example of evaluation effort deemed necessary to evaluate up to a certain Maximum Target Security Level. This table is based on previous commercial evaluation work carried out by SLAB. See also chapter 3.5, where we expand this table and describe evaluation effort expected in evaluations in the VESSEDIA project.

One Expert Day is the equivalent of the effort of one working day for a specialist in an institution equipped to carry out the evaluation. The following table, Table 5 does not refer to additional costs (neither from the side of the Evaluator or from the side of the theoretical Attacker) other than the actual required effort to carry out the evaluation. Attacks, targets, and depth are defined further in other parts of the methodology, i.e., in the definition of the scope for each evaluation (see 3.4.1). Larger targets (more areas or multiple targets) will also multiply the necessary effort.

Maximum Target Security Level	Expert Days of Evaluatiom	Execution time of Evaluation	Target of Evaluation	Depth of Evaluation
SCL 0				
SCL 1	1 day	1 week	1 sample box	Black-box
SCL 2				Black-box
SCL 3				Black-box
SCL 4	5 days	3 weeks	2 sample boxes	Black-box
SCL 5				Black-box or Grey-box
SCL 6	10 days	4 weeks	2 sample boxes	Black-box or Grey-box
SCL 7				Grey-box or White-box
SCL 8	20 days	8 weeks	2 sample boxes + open debug box	Grey-box or White-box
SCL 9	TBD	TBD	Sample boxes + Source code	White-box
SCL 10	TBD	TBD	Sample boxes + Source code	White-box

Table 5: Security Evaluation effort according to Maximum Target SCL level

2.5.2 Determining the certification level

In earlier commercial evaluation activities, SEARCH-LAB has used the definition described above successfully to relate the time of Evaluation to estimated time and cost of an attacker who seeks to attack the ToE. Our methodology has also been used successfully to communicate risks uncovered during the time of evaluation, as described in 3.6.2. It was seen again and again that the sum of risks uncovered are generally inversely proportional to required attacker effort estimated up to a certain level of protection expected.

In the following, we attempt to describe this expertise with formulas to provide a practical upper limit on certification level achieved. The actual value is named SCL (Security Certification Level) as

above. We assign an SCL value to the security of each relevant area (per Applicability Field), and the value reflects the level of security observed during the evaluation.

The final obtainable level assigned will provide an assurance up to a certain level statistically, against a certain level of attacker effort (Expert Days of an Attacker). Obtainable level is also limited by the Expert Days of the evaluation, the empirical limits observed being described in the previous chapter (Maximum Target SCL level in Table 4). The relations can be estimated as:

$$SCL_A \leq 2 * ln(1.67 * E_A) \quad (1)$$

Where E_A is the number of Expert Days of Evaluation per relevant area A for which the SCL_A value is estimated. This estimation contains constants to fit observed behaviour with a formula. We expect that the actual values might vary with different areas of expertise slightly, and that the above formula might be revised periodically.

We have also observed as a trend that the weighted sum of the Risks for the vulnerabilities found is inversely proportional to the attainable level, which we attempt to estimate as follows.

We propose a scoring system for weighting vulnerabilities based on their risk as in Table 5. For the description of assigning Risk levels for vulnerabilities found, see 3.6.2 and Table 6.

Risk [R]	M(R)
Very low	0.0625
Low	0.125
Medium	0.25
High	0.5
Very high	1
Catastrophic	2

Table 6: Proposed Risk Scoring for vulnerabilities found

Likelihood / Severity	Low	Medium	High
Low	0.0625	0.125	0.25
Medium	0.125	0.5	1
High	0.25	1	2

Table 7: Proposed $M(R_V)$ values per likelihood and severity of vulnerability

R_A is the corrected sum of risks for a given area:

$$R_A = \sum_{v=1}^{N} M(R_v) \quad (2)$$

Where there are N vulnerabilities found, each with an assigned risk value of R_v, and M is the weighting function defined above.

Then, we define D_A, which is a quantity that can be approximately described as '*expected attack days without exploits*' estimated per area:

$$D_A = E_A / max(1, 0.5 + R_A) \quad (3)$$

The above formula approximates the effect that evaluators need to cover the whole attack surface looking for vulnerabilities (proportional to E_A, the number of Expert Days of Evaluation), while it is

sufficient for an attacker to find one attack path to successfully exploit the target (approximated by D_A or '*expected attack days without exploits*').

From which we finally derive a second limit for SCL value of the area as:

$$SCL_A \leq 2 * ln(1.67 * D_A) \quad (4)$$

The above formula becomes identical to (1) when there is no finding, since R_A is then 0, and the maximum of 1 is used in (3).

When there are multiple areas covered by the evaluation, a general SCL value can be calculated as the floor of the minimum of the SCL values for areas evaluated:

$$SCL = \min_A \lfloor SCL_A \rfloor \quad (5)$$

The SCL value assigned is applicable after actual vulnerabilities uncovered were patched, and after a second check confirming this. Higher SCL values can be obtained only after a re-evaluation, usually. Practice shows that this SCL value can be used to communicate maximum security level assured, as observed in the particular evaluation.

There are of course uncertainties and statistical deviations introduced by the application of the formulas described above. Among others, we have observed the following deviations:

- Expert Days of the Evaluator cannot be directly related to the Expert Days of a hypothetical attacker using universal constants: e.g., the evaluation methodology (white- / grey- / black-box) and the level of test automation (thus, number of tests executed) will directly influence the relation between the two. The above formulas describe current estimation using the constraints observed in Table 4 and taken into account in chapter 3.5.
- Too few Expert Days of evaluation lead to larger relative errors in the estimated attacker effort and expressed SCL value, thus the assurance provided is heavily limited (i.e., 1 day of evaluation is usually sufficient to estimate up to SCL 1 value only). Thus, in practice, SCL levels below value of 4 might be considered as informative only, with limited assurance.
- Vulnerabilities uncovered cannot always be combined into exploits that implement practical attacks, thus even a large number of vulnerabilities might not lead to practically exploitable vulnerabilities in the ToE. When assigning higher SCL values, it is advisable to carefully evaluate combination of vulnerabilities uncovered, and study feasible attack paths when assigning risk values. In practice, Likelihood value can be used to reflect uncertainties due to other vulnerabilities in the same evaluation.
- Low-risk vulnerabilities may or may not have actual practical impact on the security of the ToE, depending on the design (e.g. security measures present might provide effective countermeasures against whole attack classes). Still, they contribute to a general (design- and code-) quality impression, and when there are many low-risk vulnerabilities, they show the potential for exploitability generally, in a statistical sense.
- In the application of formula (3), the use of the constant of 0.5 is necessary for cases when there are few vulnerabilities with lower-level risks. With this constant, one vulnerability of High risk does not decrease the SCL_A value compared to the case when there was no vulnerability found (due to the use of minimum denominator of 1). Although there is an uncertainty visible here, the resulting SCL values are mostly in line with our expertise.
- The SCL levels attainable cannot be raised arbitrarily by extending the evaluation effort – there are diminishing returns for the increase of efforts. The formulas reflect this to a certain level (thus the use of logarithm), but in practice, there is also a natural hard limit depending on several factors: evaluation methodology, level of expertise available, practicality of further test cases, costs, and also the security level of the ToE. Thus, we advise to apply this certification scheme in VESSEDIA only for the lower- to medium-critical devices. Higher-

security devices might need to be reviewed and certified more systematically, e.g. according to Common Criteria as well, see also 2.5.4.

A hypothetic example to determine SCL value would be:

- 10 days of testing executed.
- Associated limit on SCL_T would be: 2*ln(1.67*10)=5.63. This is the Maximum Target SCL Level attainable.
- In total, 21 vulnerabilities were found with the following Risk values, and their contribution to the weighted sum calculated:
  - Catastrophic: 1*2=2
  - Very high: 3*1 =3
  - High: 5*0.5 = 2.5
  - Medium: 2*0.25 = 0.5
  - Low: 4*0.125 = 0.5
  - Very low: 6*0.0625 = 0.375
- Sum: R_A = 2+3+2.5+0.5+0.5+0.375 = 8.875
- Corrected days: D_A = 10/(8.875+0.5) = 1.067
- Obtainable level: SCL_A = 2*ln(1.67*1.067) = 1.155
- Total score of SCL_A is limited by 1.155 due to the vulnerabilities uncovered
- Further areas can be calculated separately, but if only this area was present, the assigned level would be SCL=1

Further examples showing the effect of various sets of vulnerabilities:

- 1 catastrophic vulnerability in a 10 days evaluation:
  - D_A = 10/2.5 = 4, which leads to SCL=3 (SCL_A=3.798)
- 1 catastrophic vulnerability in a 20 days evaluation:
  - D_A = 20/2.5 = 8, leading to SCL=5 (SCL_A=5.185)
- To reach SCL=4 in a 10-day evaluation, the following vulnerabilities and risks are allowed:
  - At most 1 High-risk vulnerability or 3 Medium-risk vulnerabilities still result in maximal level, which is SCL=5
  - 1 vulnerability with Very high risk (SCL_A=4.82)
  - At most 3 vulnerabilities with High risk (SCL_A=4.245)
  - At most 7 vulnerabilities with Medium risk (SCL_A=4.009)
  - 1 High-risk, 4 Medium-risk, and 2 Low-risk vulnerabilities (SCL_A=4.009)

2.5.3 SCL certification process

For each Application Area identified (see Chapter 3.1), Security Objectives are collected during the Information gathering phase, and Threats are derived during the Threat modelling phase of the Evaluation methodology (Chapters 3.2 and 3.3, respectively).

Feasible test cases are collected from various sources (based on state-of-the-art knowledge collection from commercial evaluations) related to security objectives and threats. Each test case is also annotated by a corresponding SCL level value.

In the context of VESSEDIA, the project goal is also to evaluate the use of VESSEDIA tools and metrics provided by them (see Chapter 4 for a list of tools and D4.1 [3] for the description of metrics), and by carrying out the evaluation, it is expected that concrete measurable results will be generated for other tasks inside WP4 regarding the feasibility of using the tools in different levels of certifications.

2.5.4 Connection to Common Criteria

VESSEDIA project has connections to common criteria in different ways. From D1.2, two of the use-cases (namely INRIA's Contiki and the DA's Aircraft Maintenance System, not CEA's 6LowPAN management platform) list the security requirements following CC standards.

In WP6:D6.4, we can establish connections with the candidate ISO standard. Our candidate ISO standard is prepared as one of the series of single tool capabilities which are used with ISO/IEC 20741 "Systems and Software Engineering- Guideline for the evaluation and selection of software engineering tools".

The "Verified in Europe" label is more than a by-product and does not depend on the success of the ISO standard candidate. Guidance given along the "Verified in Europe" certification allows to strengthen software's safety and security capabilities.

As from the DoA, WP4:T4.4 "Contribution to Common Criteria process" will show "how the methodologies and tools developed in this project may contribute to Common Criteria process of safety and security critical applications". We will in this task, define how the new items will potentially support the CC certification process, especially when reaching for the levels EAL4+ and above. The study of the certification process should produce documents that are suitable for standardization process. This task makes it sure that the documents provide suitable foundation for standardization.

We will study what extra support can be brought by the VESSEDIA tools and methodology to the CC activities and, when possible, implement extra functionalities to do so. The study in D4.5 [5] will be based on the use-cases of WP5 whose security evaluation process will be considered as a basis.

More generally, in scientific literature, formal verification connections to CC are sometimes discussed. For instance, in [26], it is shown that:

> *[...] it is theoretically possible to use code-level specifications as development models in the Common Criteria. However, they are not required, and whether they are adequate in real- world certification will depend on the evaluator. The standard itself tends to be very vague on how formal methods are used. In our opinion it is also possible to replace functional testing by formal verification altogether even if verification is not required explicitly.*
>
> *While it is clear that most security policies are only expressible at a high level of abstraction, they are only enforceable if there is a sound trail of evidence, tracing from the abstract-level specification to the implementation. If the principal goal of the CC for high-assurance reliably secure software is to be reached, certification has to include state-of-the-art verification techniques.*
>
> *We are not aiming at the creation of an even higher assurance level beyond EAL 7. This would bear the risk of even lowering the overall acceptance of formal verification in the industry. Statistics show that of over a thousand certified products, the vast majority has reached EAL 4+ or below, while EAL 5 is at least quite common for smart cards and their applications. Only three hardware devices and no software products were awarded EAL 6 or higher until now. This shows that hopes for a broader usage of formal methods through means of the Common Criteria were not fulfilled. And even in academic contexts, there are few results. As explained in the work by Chetali and Nguyen, reaching an actual certification is far more work-intensive, and verification is just a fraction of that. In the end, they reached only EAL 4+. A full EAL 7 evaluation would also require other laborious tasks such as integration and penetration tests.*

These discussions identify clearly the concerns and difficulties met when attempting to make use of formal approaches in CC contexts. But they emphasize also the needs for adaptations, ideally in both ways (formal towards CC, and CC towards formal), and justify the efforts developed in VESSEDIA project.

Chapter 3 Security evaluation methodology in VESSEDIA project

This chapter describes the Security Evaluation methodology that will be used as the basis for VESSEDIA approach for security evaluation. This methodology was developed by SEARCH-LAB Ltd. (SLAB) and was used in commercial security evaluation projects successfully for several years. The first public description of the MEFORMA methodology [9] was developed in nSHIELD[2] project, which was a project co-funded by the ECSEL (former ARTEMIS) JOINT UNDERTAKING (Sub-programme SP6).

The methodology was subsequently applied in the FP7-funded STANCE project with success [18].

The methodology is being further developed and extended with methodology to assign Security Certification Levels to the evaluated targets as part of the assessment results – see also Chapter 2.4. The description herein defines the state-of-the-art version of methodology intended to be applied in VESSEDIA and to be used in further WP4 deliverables (i.e., [7]) as well. Comparison to other evaluation frameworks is described in 2.3.

This evaluation methodology is developed as a scheme for certification of devices referred to as “lower critical devices” in the VESSEDIA project (see in 2.4), and complements formal verification according to Common Criteria (CC, descibed in 2.5.4), in the sense as it was declared in ENISA position paper as **a baseline security certification** (essentially a “lightweight” certification) for ICT products – see 2.3.1.3.

3.1 Definition of the Applicability Field (Scope definition)

The first step of any evaluation is the identification of the ToE (system to be evaluated), and specifying the scope of the evaluation based on the needs of the Developer and the expertise of the Evaluator. Developer and Evaluator roles are impersonated by consortium members responsible for the functionality and by WP4 partners, respectively. The approach implies that the work is accomplished in different phases that build upon each other, and that each phase ends with related deliverables documenting the results.

In the case of the VESSEDIA evaluations, the scope was defined based on the use case descriptions from WP5 as well as discussion with the use case partners. In this first iteration of evaluations, the scope would need to be limited to certain open-source software that the use cases build on.

An important aspect of this step was establishing the ToE itself and ensuring that it remained fixed throughout the entire evaluation – this was done in the STANCE project by use case partners providing virtual machine deployments of the relevant components.

[2] http://www.newshield.eu/what-is-nshield/

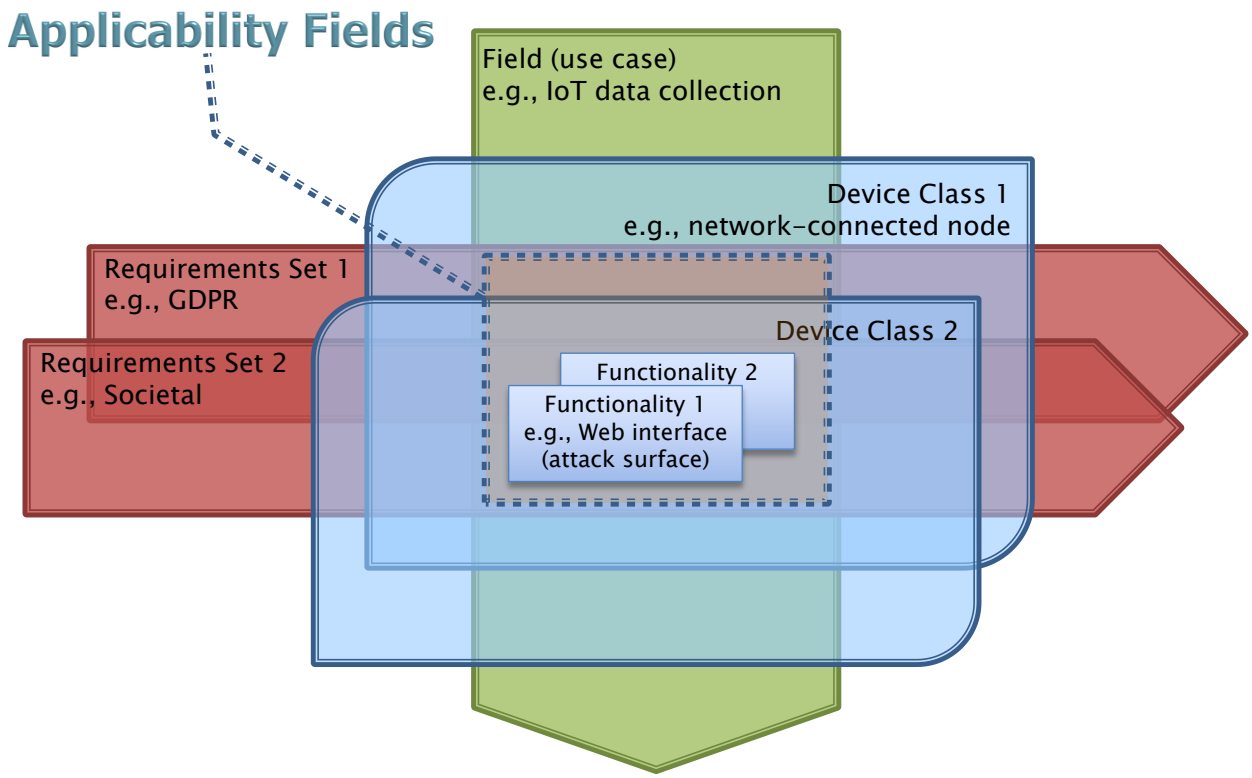

Figure 3: Definition of Applicability Field for Evaluation Plans

Applicability Field is an abstract set based on the definition of Security Requirements, Security Objectives, and Valuable Assets.

The **Applicability Field** is defined via the following:

- List of **Valuable Assets** for the field
- **Security Requirements** considered relevant (e.g.: Requirement Set for GDPR)
- **Security Objectives**
- **Threat Modeling** derived from the above, along with a **Risk Analysis**
  - Each **Threat** identified is evaluated based on its **Likelihood** and **Severity**
  - **Threats** are categorized against their relevant **SCL level** respectively
  - In an evaluation the actual **list of threats** (with their **Likelihood** and **Severity**) is assessed
- The description of the **Applicability Field** is via the following two documents:
  - **Evaluation Plan** describing the above items
  - **Test Case Specification** describing the relevant test cases versus applicable SCL level
  - **Developers' Guide** describing security requirements and best practice

Several elements of the above are already studied and described D1.2 [2] for WP5 use cases

A typical MEFORMA evaluation project consists of the following steps:

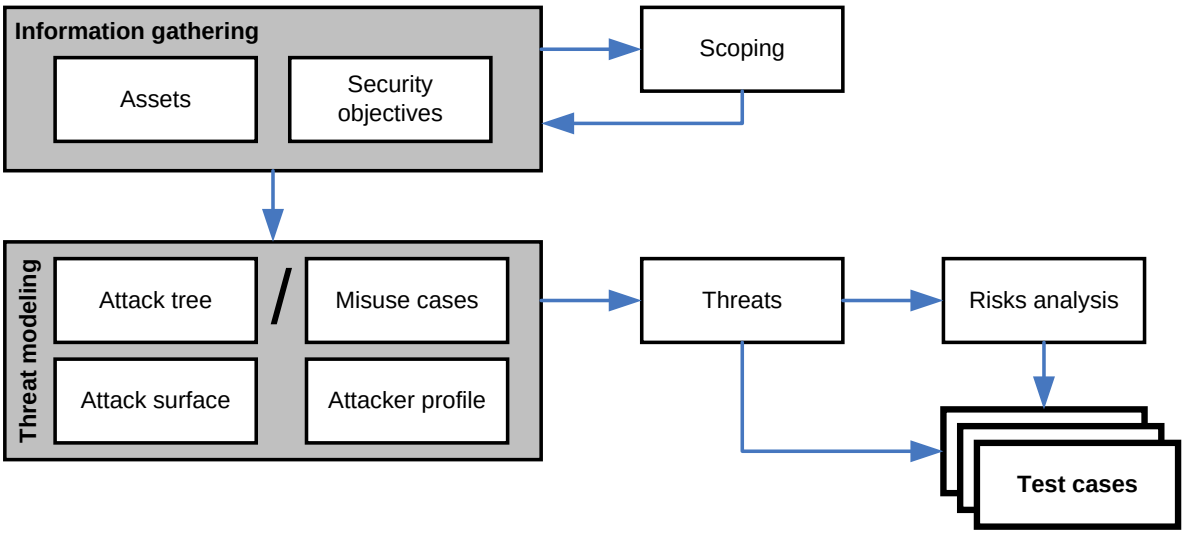

Figure 4: Preparatory steps for defining the Evaluation Plan for the Applicability Field

3.2 Information gathering - identification of security objectives

In information systems, providing *security* covers several aspects, such as defending information from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. Security objectives are the high-level requirements and goals that are most important to the stakeholders and to the requirements that must be met to comply with relevant legislation, policies and standards [15].

Offering these generic *security objectives* is difficult, as attackers can, in almost all software today, assume the presence of vulnerabilities which can be exploited [16]. Vulnerable software can be invaded, locally and remotely, and modified to cause damage. Malicious code can replicate itself and travel to other systems, often undetected by users.

The main goal of this step is to specify the security objectives. For this, one should first identify and understand the valuable assets within the system that need to be protected, and then for each asset determine which of the independent security objectives (typically taken from the CIA triad i.e. Confidentiality, Integrity, and Availability) are relevant. The assets are then further grouped into categories appropriate to the specific evaluation – for instance, a software evaluation would likely have ‘software assets’, ‘data assets’, and ‘other assets’ categories.

The work of WP1 defined the relevant security properties in D1.1 [1] and detailed the security objectives for several use cases in D1.2 [2].

3.3 Threat modelling

Once the elements of the system are identified and understood, threat modelling is accomplished according to the security objectives. This can be done by various means and following various approaches; we mostly use the following:

- **Attack tree** (Schneier, 1999) modelling consists of conceptual diagrams of perceived threats to a system. This process is performed by identifying an attacker goal, and then modelling the various ways these goals can be achieved.
- **Misuse cases** are similar to the use case UML formalism, but instead of describing ways to use system functionality, they present ways on how to misuse it (Alexander, 2002). The misuse cases are derived from the normal use cases: for instance, for a normal ‘shutdown’ use case there may be several misuse cases defined where the system can be shut down.

In **threat modelling**, as security testing is about how a system should not work, there is a remarkable emphasis on the evaluators' security experience, i.e. its knowledge about how things can go wrong, and also about the attack trends among the attackers. For this, one can rely not only on its experience, but can also refer to repositories and knowledge bases, like the ENISA Threat landscape (ENISA, 2012) and further publications.

Optionally, **attacker profiling** may also be performed in this stage. This identifies several different types of attackers that may have different goals when it comes to attacking the security of the ToE, and may also have different resources and expertise at their disposal. Example profiles are insider, exploiter, misuser, and thief.

The end result of this step is a set of **threats**.

3.4 Evaluation Plan and Test case specification

3.4.1 Evaluation Plan for the Applicability Field

A public technical document, which is containing:

- Description of **Applicability Field**:
    - Composes a set of **Functionalities** together with relevant **Device Classes** providing the functionality
    - Constrained to relevant **Use Case** and **Requirements Set**
    - Collects relevant **Assets**, **Security Objectives**, **Threats**
    - **Security Objectives** implicitly define the attacks that are considered relevant
- Defines **Security Evaluation Methodology** related to **SCL** for a certain **Applicability Field**
- Encompasses **Use Case** and relevant **Requirements Set**(s)
- Describes list of **Risks**
- Contains **Threat Modeling**, **Risk Analysis**
- Leads to **Test Case Specification**, usually attached as an internal/confidential description, which lists required test categories and methods

3.4.2 Test Case Specification

This is usually an internal technical attachment to the **Evaluation Plan** containing:

- Describing exact **Test Methodology** according to **Evaluation Plan**
- Describing detailed **Evaluation Plan** according to **Assets**, **Risks**, **and Threats,** …
    - Automated vs. manual analysis steps, tools used
    - Black box / white box
    - Includes design review, source code analysis, etc. or not...
    - Includes formal and code analysis methods depending on the use cases and target SCL Level
    - Process-related tests? (e.g., use of versioning)
- Specifies test requirements vs. **SCL Level**, containing a **Test Case Matrix** – see 3.4.3

The goal of the VESSEDIA project is to develop and evaluate VESSEDIA tools that provide valuable results in the evaluation. Test Case Specification will consider VESSEDIA tools available (see Chapter 4). Test Methodology will provide exemplary test actions for the use of tools, while test steps will detail usage.

Test Case Specification will provide a step-by-step guide to carry out the evaluation itself, which will be described in VESSEDIA deliverable D4.6 [7].

3.4.3 Connection to Security Certification Levels (SCL)

During threat modelling, many potential threats will be identified. Some of these threats may be considered out-of-scope for the evaluation due to some reasons (for instance, being unlikely or out of the control of the Developer, like the vendor's secret key being leaked). However, most of the threats will require investigation to confirm their feasibility – which is the main goal of the evaluation.

To that end, the evaluators categorize the relevant threats, and accomplish a preliminary risk analysis to reveal which are the most important threats. This prioritization is especially important, since the project size and the allocated efforts might not be in line with the volume of the actually revealed and relevant threats. Risk analysis shows which will be the issues that should be put in focus and included in the evaluation, and which can be possibly omitted due to being less critical or beyond practical resource limits.

The definition of SCL levels (see 2.5.1) is a tool aimed at providing further means prioritizing the actual test cases considered relevant with regard to the evaluation effort deemed feasible, and the corresponding Attacker effort and cost estimated for threats.

The actual SCL values assigned to individual test cases are described in a Test Case Matrix, based on prior knowledge and expertise. Costs of the execution of certain test cases can be considered when planning an evaluation, in line with the depth and run-time of the evaluation for the corresponding SCL level (see Table 3 in Chapter 2.5.1). See also 2.5.2 for calculation of the SCL value.

Several of the VESSEDIA metrics defined in D4.1 [3] would be suitable to measure the complexity of functions under test. We will study whether these metrics can be used in correlating to the complexity of the evaluation and to the SCL value to be assigned.

3.4.4 Developers' Guide for Applicability Field

This public document is aimed at the developers of the Functionalities implemented, and contains:

- Security requirements design and implementation,
- Secure Coding best practices,
- Software Development Life Cycle processes.

3.5 Evaluation

Building upon previously described preparatory work, the evaluation means the execution of the test cases already specified. Actual evaluation of a test case can consist of black-box / grey-box / white-box testing or source code review, and can also include reverse-engineering of the system. Manual execution of test cases can be mixed with automated evaluation, for instance fuzzing or penetration testing tools.

In the case of VESSEDIA project, we will define test cases built on the VESSEDIA tools described in Chapter 4. Outputs of tests will be included in the Evaluation Report, and the evaluation process itself will be assessed as to rate the usefulness of tools and metrics according to D4.3 [4] in the following WP4 documents: D4.5 [6], and D4.6 [7]. The next step of the work is to define the Evaluation plan according to this document, selecting the actual ToE from the WP5 components (based on outcomes defined in D5.1, D5.3, and D5.5). The detailed description of the Evaluation process for the selected targets will be defined according to a Maximum Target SCL Level, which is selected based on the target's security criticality, and available effort.

The executed test cases during an evaluation depend on the Maximum Target SCL Level. In the following table, we summarize the planned characteristics of the evaluation for each SCL level. For

each level in the table below, the evaluation techniques used include all techniques described in the previous levels, however, only the new techniques and modifications are defined.

Intermediate SCL levels (e.g. SCL 3) are currently not specified as target, but they can be achieved as the result of the evaluation, according to the methodology described in 2.5.2. The fine granularity of the SCL level scale makes it possible that the results can be evaluated more precisely, even if only fewer target levels are used.

Maximum Target Security Level	Execution Time of Evaluation	Target of Evaluation	Depth of Evaluation	Evaluation Activities
SCL 0	n/a	n/a	n/a	SCL 0 means that the product failed in the evaluation, so SCL 0 evaluation will not be performed in practice.
SCL 2	1-2 weeks	1 sample box	Black-box	Using automated tools to find known or common vulnerabilities, such as CSRF, XSS, command injection.
SCL 4	3 weeks	2 sample boxes	Black-box	The evaluation uses a systematic approach (such as MEFORMA [9]) to maximize the test coverage. Techniques such as reverse engineering, basic fuzzing and hardware tests can be used.
SCL 6	4 weeks	2 sample boxes	Grey-box	Chain of trust analysis, advanced fuzzing and more detailed hardware tests are introduced compared to the previous levels. From this level, the evaluation requires certificate or evaluation of trusted components, such as hardware elements and embedded software components.
SCL 8	8 weeks	2 sample boxes + open debug box + source code of most critical parts	White-box	Source code verification is required for the most critical parts of the applications (modules) using formal or semi-formal analysis, such as described in WP2 and WP3.
SCL 10	12 weeks+	Sample boxes + Source code	White-box	The previous level is extended with the verification of the integrated system using semi-formal and formal source code analysis methods, such as described in WP2 and WP3. In addition, the underlying OS should be certified as well.

Table 8: Evaluation guidelines in case of different target SCL levels in VESSEDIA project

Usually, due to the substantial extra effort involved in exploiting a vulnerability, the Evaluator will always focus on discovering and documenting previously unknown vulnerabilities first; vulnerabilities are only exploited if explicitly requested by the Developer (such as to demonstrate the consequences of a particularly dangerous vulnerability).

3.6 Documentation and recommendations

As a result of the evaluation, a list of verified threats is compiled in the form of the list of findings. Most of the time, these are threats that were already identified during the threat modelling step, but completely new threats can also surface during the evaluation.

When documenting the threats, we use several symbols to denote the results of individual tests within a test case. These symbols are as follows:

- ✓ **Normal operation.** The outcome of the test indicates that the implementation is correct; no findings.
- 💣 **Problem.** The outcome of the test has clearly identified a security problem.
- 😐 **Potential / possible problem.** The outcome of the test does not clearly indicate a security problem, but may lead to unexpected or abnormal operation. This symbol is also used if a security issue is suspected, but could not be verified.

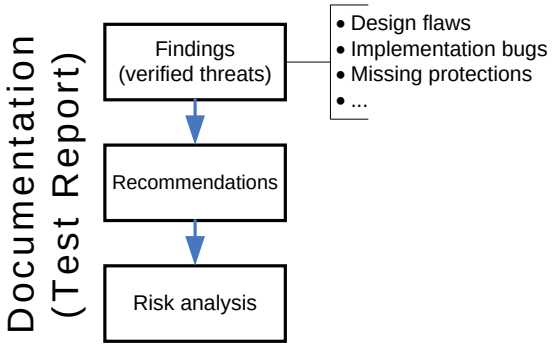

Figure 5: Documenting the findings

For each verified threat, recommendations are given for techniques that could be used to completely eliminate the threat, or at least reduce the associated risks. In each case, the goal is to reduce either the likelihood or the severity of the threat.

All results are contained in an Evaluation Report delivered to the Developer, who should make the appropriate steps to address the issues found.

3.6.1 Conformance to requirements

Additionally to the threats uncovered, conformity to security objectives as defined in the Applicability field (see also 2.4 for a list of security objectives for common IoT assets) will be evaluated.

Higher-level security functional and assurance requirements are specified for the potential ToEs in other VESSEDIA deliverables such as D1.2 Requirements descriptions for WP5 use cases [2]. The conformance to specified requirements can be verified according to other methods, like Common Criteria.

3.6.2 Risk analysis

The risk each discovered threat poses to the system is evaluated. This is done by specifying the severity (damage that can be potentially done by realizing the threat) and the likelihood of each threat (the difficulty of realizing the threat). This latter can be dependent of many factors, including the needed resources (time, money, tool accessibility, etc.) and the expertise level required by an attacker to realize the threat, i.e. commit a relevant attack.

The risk is the product of these two values using the standard likelihood X severity risk calculation:

Likelihood / Severity	Low	Medium	High
Low	Very Low	Low	Medium
Medium	Low	High	Very High
High	Medium	Very High	Catastrophic

Table 9: Likelihood × severity risk calculation.

The severity level corresponds to the security requirements described in the applicability field. As an example, the severity can be defined with the following levels:

- **Low (L)**: Vulnerabilities that cannot be exploited or can only result in unexpected (functional) errors. Minor data leakage, user misleads or transient denial-of-service type attack.
- **Medium (M)**: Leakage of confidential information or unwarranted access to system resources. Permanent denial-of-service type attacks against single device.
- **High (H)**: Subversion of system components or code execution. Transient denial-of-service type attack against multiple devices.

We categorized the likelihood with the following levels:

- **Low (L)**: Rare events. The attacker needs detailed knowledge about the system, or needs special equipment. Some of these events may only be performed with the help of an insider. The attack can be performed from the local network after authentication.
- **Medium (M)**: The event may happen. The attacker only needs normal knowledge about the system and the attack can be performed with normally available equipment. The attack can be performed from the local network without authentication or remotely after authentication.
- **High (H)**: The event occurs quite often. The attacker only needs minor knowledge about the system and does not need any additional equipment. The event can occur due to wrong or careless usage. The attack can be performed remotely without authentication.

The risk value of each threat can take the following levels:

- **Very Low (VL):** The threat has a very minor – but still not negligible – effect on the security of the asset.
- **Low (L):** The threat has a minor effect on the security of the asset.
- **Medium (M):** The threat has a noticeable effect on the security of the asset.
- **High (H):** The threat significantly endangers the asset.
- **Very high (VH):** The threat significantly endangers the asset or the system as a whole.
- **Catastrophic (C):** The threat presents a critical risk to the system as a whole; if not mitigated, its effects could put the entire business process at risk.

In our current evaluation work, we will also associate the risk of each threat found with the Security Certification Level associated, based on its estimated value – see Chapter 2.5.1.

3.7 Review

In usual evaluation scenarios, a several-weeks-long review phase is usually reserved for the Developer to review the results and fix the revealed weaknesses following the evaluation. At the end of this period, the Evaluator receives a new version of the ToE, and re-runs the relevant test cases on it to verify if the threats have been appropriately tackled. A residual risk analysis is accomplished showing the threats that still remain in the system after the review. Any new threats discovered during the review would also be presented in the Review Report.

Chapter 4 VESSEDIA tools in the security evaluation

VESSEDIA tools that will be developed by project partners and also used in the security evaluations are enumerated in this chapter.

4.1 Frama-C framework

Frama-C is an open-source C and C++ source code analysis framework. It is built on a kernel and contains numerous open-source and proprietary plugins.

The Frama-C components share 1) the CIL (C Intermediate Language) representation of the source code that represents an internal abstract syntax tree, and 2) the ACSL (ANSI/ISO C Specification Language) specification language and its extensions for C++ also called ACSL++. The later covers a subset of C++ and is work in progress.

The Frama-C plugins that we generally use (in R&D only context once again) are as follows:

- Evolved Value analysis (EVA)
- WP for deductive verification
- Scope & Data-flow browsing
- Slicing
- PathCrawler (just assessed, not used currently)
- E-ACSL (executable ACSL for dynamic analysis)
- Metrics
- Frama-Clang (prototype).

Some other experimental plugins have been developed, namely:

- RPP
- SecureFlow
- Report
- MDR
- FlinderSCA

We will describe these plugins below in more details.

4.2 Fuzzing with AFL test generation tool

During the STANCE project, SEARCH-LAB developed Flinder and FlinderSCA plugins to Frama-C to extend the static code analysis capabilities of Frama-C with security and fuzz testing [17]. Its main purpose was to verify alarms reported by static analysis plug-ins such as EVA, and – if a positive test result is observed – notify the plugins of the fact in order to mark the alarm as real. For a brief description of the structure of Flinder and FlinderSCA, see also section 5.2.

In the VESSEDIA project, a next generation of the fuzz testing plugins will be researched and developed. The Flinder fuzz testing framework is abandoned in favour of AFL (American Fuzzy Lop - http://lcamtuf.coredump.cx/afl/), which is a state-of-the-art fuzzer providing best in class results employing a novel type of compile-time instrumentation and genetic algorithms to automatically discover clean, interesting test cases that trigger new internal states in the targeted binary.

To make integration of AFL fuzzing with Frama-C possible, a new plugin based on FlinderSCA will be developed, which will interface AFL this time, called AFL-SCA.

The structure and operation of the AFL-SCA plugin will largely be similar to the operation of FlinderSCA, integrating several of the components existing in the Frama-C environment: SANTE/StaDy, the AST representation, and the Slicing plugin to generate an instrumented code for fuzzing by AFL. The results of the fuzzing (vulnerable code paths) will be reported back to the Frama-C environment.

4.3 FlowGuard monitoring tool

Input: C/C++ source file or files.

What it is: It is a gcc plugin (which can be invoked by adding an additional flag for the compilation/linking phase)

It instruments/hardens the resultant binary with the Data Flow Integrity. It also generates a .so binary that will be able to check the DFI inside the binary.

This is in the case of user-space applications.

In the case of kernel-space apps/libraries the .so is not an option (due to kernel protection mechanisms), but we're currently working in a solution that will use a kernel module (since in this case it is to enforce parts of the kernel or the whole kernel, which makes perfect sense) but this will take a while because kernel internals are even harder that gcc internals.

Output:

User-land (done without optimizations but complete DFI): a binary enforced with the static DFI inside and a .so to perform the runtime integrity checking of the DFI avoiding code reuse attacks (and easier ones too).

Kernel-land (in progress): the targeted kernel binaries and, instead of .so for the runtime checking a kernel module (again, this is future work).

For a more comprehensive view of the process and gcc internals, please see the slides presented at the Technical Meeting in Paris about CFI and DFI for runtime protection [27].

4.4 Diversity symbolic execution tool

DIVERSITY is a customizable model analysis tool based on symbolic execution. It is an open source tool (Eclipse license EPL) available in the Eclipse Formal Modeling Project (follow the link below). DIVERSITY provides a pivot language called xLIA (executable Language for Interaction and Architecture) introducing a set of communication and execution primitives allowing one to encode a wide class of dynamic model semantics, e.g., hierarchical timed communicating Symbolic Transition Systems STS , UML/SysML (UML State Machine, Sequence Diagrams,…), SDL, and abstractions of hybrid systems. DIVERSITY is an extensible tool allowing customizing the basic symbolic treatments to implement specific Formal Analysis Modules -FAM for short- (e.g., Model-based Testing -MBT for short- algorithms, exploration strategies and heuristics, etc.). DIVERSITY is connected with SMT solvers, e.g., CVC4, Z3 and YICES which can be easily used to implement new FAMs. DIVERSITY is composed of two distinct parts depicted in the following Figure: an Eclipse-based Graphical User Interface (GUI) and an SE kernel developed in C++.

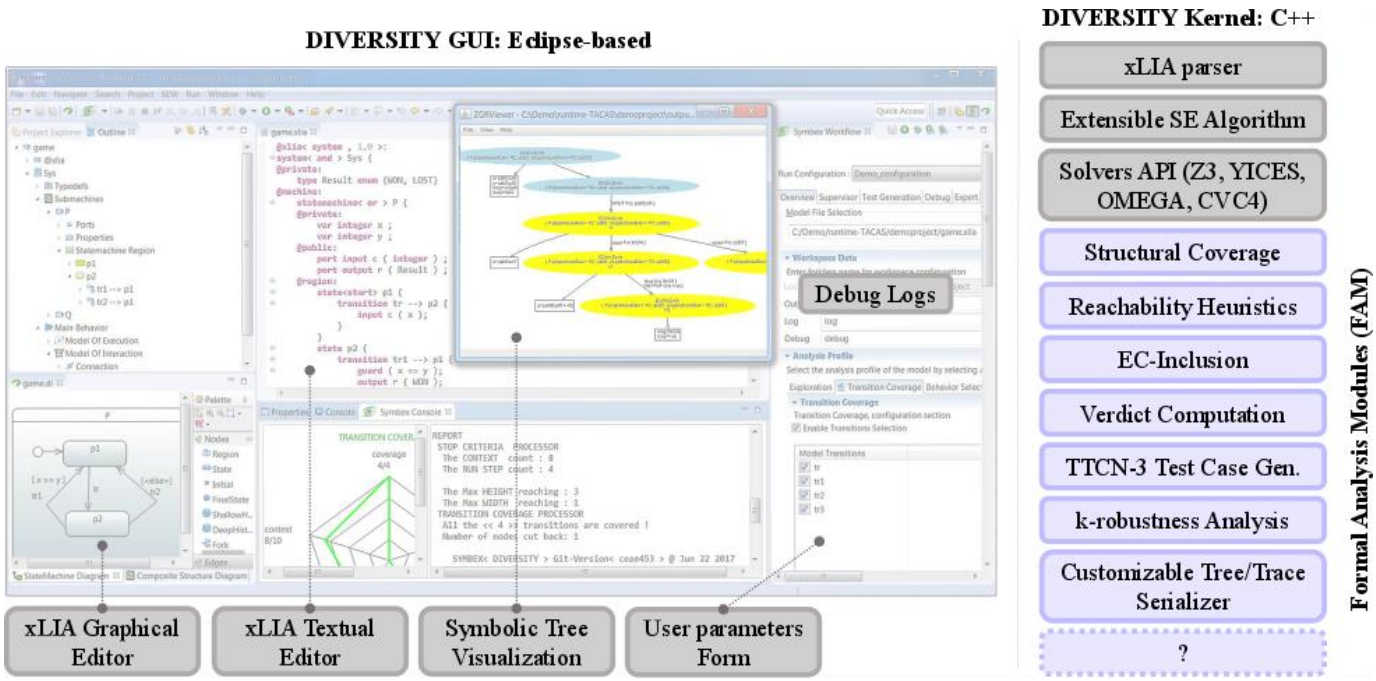

Figure 6: GUI of Diversity tool

It provides a textual editor and a graphical editor for xLIA models. The latter is implemented as a UML/SysML specialisation using the Papyrus technology. The GUI provides as well utilities for symbolic tree visualisation and debug with different logging levels. Some of the existing FAMs of DIVERSITY have been associated with GUI forms in order to enter required user parameters. Note that DIVERSITY kernel EXE can be run in command line: it takes as input an XML-like user parameters file including the xLIA model location.

These are descriptions of some available FAMs which will be used and assessed in the frame of the VESSEDIA project:

- EC-Inclusion: This FAM stops the symbolic execution of any EC (at pre-filtering) when it is included in another already computed one in the symbolic tree
- Reachability heuristics: This FAM computes a set of symbolic states targets of some paths satisfying a coverage goal such as covering (successively) some transitions, states, I/O actions, function calls, or satisfying logical formulas on the model state variables
- k-robustness analysis: this FAM is used to analyze the robustness of a design to some known cyber-attacks, in terms of intrusion detection. The goal is to ensure that the design model is equipped with a watchdog able to emit a warning in less than k actions of the attacker (k is an integer specified by the designer of the detection mechanism endowed in the watchdog).

Beyond these few examples, one major application domain for DIVERSITY is model-based testing. Let us emphasize that the development of DIVERSITY has been mainly driven by needs issued from model-based testing, such as time modeling, customizable tool or selection criteria …

In the scope of VESSEDIA, a new FAM is being developed in DIVERSITY: its objective is to be used intertwined with the Reachability heuristics in order to infer contracts for a cooperation of function calls which implement specific safe high level system scenarios.

https://projects.eclipse.org/proposals/eclipse-formal-modeling-project

4.5 Papyrus modeling tool

Papyrus is an UML standard Open Source modeling tool for embedded systems' design and specification and especially real time critical systems. Papyrus will take the implementation process to industrialization and will accompany it through the whole process until a high quality level of software development. Distributed all around the world, Papyrus is Eclipse foundation UML2 graphic modeling tool (190 members in the world including IBM, Google, Oracle and SAP)

In the context of VESSEDIA, Papyrus brings two main contributions:

- The modelling of security and software requirements and properties at different levels of abstraction, through a well-delimited ISO 42010 Architecture Description Language (ADL).
- Tools to bridge the gap and enforce traceability between higher level requirements and properties and their lower-level refined requirements and properties. Concretely we propose to bridge the gap between textural requirements to code properties.

As a reminder, the ADL is an aggregation of subsets of existing modelling languages. As such, Papyrus will offer editors for the following subsets of modelling and specification languages: SysML-Sec requirements, UML structured class, state-machine, and interaction elements, ACSL. These modelling languages allow the software and security engineer to specify the software/security requirements and properties, and progressively refine them until we have code-level properties. The following requirements/properties, at different levels of abstraction, are considered:

- System design level requirements and architecture
- Software design level architecture, interactions, and function properties
- Code implementation level relational properties, function and function body properties

Although the modelling and specification of requirements and properties are a challenge itself, tracing the different requirements and properties, at different levels of abstraction, is just as important. Indeed, refined properties must be consistent with higher-level requirements. Therefore, Papyrus will also be used to partially automate such refinement processes. To accomplish such a feature, we propose transformation and inference tools. Within VESSEDIA, the following tools will be developed:

- Traceability tool: This tool will ease the designer's task of manually tracing refined requirements and properties, such as UML interactions, to a higher-level requirement and properties, such as SysML-Sec textual requirements.
- Inference tool: This tool will take as input requirements and properties modelled as UML interactions, and automatically infer the program relational properties in ACSL, that have to be valid to ensure the constraints specified in such interactions.
- Code generation tool: This tool will take as input software architectures with function pre/post-conditions written in ACSL and generate the equivalent C/C++ code with the ACSL annotations.

4.6 SecuRate plugin and service

SecuRate metrics is aimed to measure the criticality of an alarm in terms of the estimated number of affected products and the estimated number of affected devices. In general, a complex software solution uses a lot of libraries, reused codes and open source software projects. In case of the IoT-related code, the code reuse is also an increasing trend, since it is required to implement complex functionality within a very limited time and resources.

By using SecuRate plugin, the evaluator can focus on components common in most projects and also will receive information about known vulnerabilities in the evaluated component.

See D4.1 [3] for a detailed description of the implementation and operation of the plugin.

4.7 VeriFast verification tool

Input: C or Java source code files, with VeriFast annotations included as special comments (similar to how Frama-C takes annotated C source code files).
Output: Either "0 errors found", which guarantees absence of certain kinds of programming errors, or a symbolic execution trace leading to a verification failure, shown inside the VeriFast Integrated Development Environment, which allows the user to step through the trace and inspect the source location and the symbolic state corresponding to each symbolic execution step.

Documentation : see https://github.com/verifast/verifast#documentation.

4.8 EVA plugin

The EVA plugin is the successor of the Value plugin of Frama-C and performs an Abstract Interpretation of the source code provided as input to deliver a set of potential run-time errors.

Once properly configured, EVA runs automatically. Contrary to many AI analysis tools, EVA is exhaustive and guarantees to produce the complete list of potential run-time errors. If an alarm is not emitted for an operation in the source code, then this operation is guaranteed not to cause a run-time error.

EVA is modular and uses several abstract domains, namely: 1) sets of integers (enumerations, intervals and intervals with periodicity), 2) floating-point values and intervals, 3) sets of addresses for dynamic objects, and 4) an imprecise mix of addresses.

See http://frama-c.com/value.html.

See also chapter 5.4 Frama-C Value plugin.

4.9 WP plugin

The WP plugin is the successor of the Jessie plugin of Frama-C and implements a Hoare calculus in order to prove the correctness of some source code w.r.t. its ACSL specifications.

The WP plugin is semi-automatic as it requires users to write complete specifications in ACSL. The WP plugin produces a set of Verification Conditions (VC) that will be discharged by predicates simplifiers, *automated theorem provers* like Alt-Ergo or *interactive proof assistants* like Coq. Other provers are also supported *via* the Why platform (see http://why.lri.fr/) used by Jessie. The complete proof of the VC generated guarantees that the source code is compliant to its specifications.

See http://frama-c.com/wp.html.

4.10 Metrics plugin

The metrics plugin of Frama-C is a simple plugin that calculates automatically some measurements on the input source code, namely

- McCabe's cyclomatic complexity.
- Halstead complexity.
- EVA coverage estimate (once EVA has been run on the code under analysis)

See http://frama-c.com/metrics.html.

4.11 Report plugin

The WP plug-in can export statistics on generated proof obligations. These statistics are called WP reports and are distinct from those property reports generated by the Report plugin. Report is a companion plugin to the main analysis plugins producing a summary about properties to be proven and their status, including their dependencies. Namely, a plug-in can mark a property as validated under the assumption that another ACSL annotation holds. Until the latter is itself marked as valid, the former will be reported as “Valid under hypotheses” instead of “Valid”. The report plug-in has several filtering capabilities and provides either a textual output or a table in CSV format, suitable for drawing statistics, such as observing the growth of validated properties during a development.

4.12 MDR plugin

The MarkDown Report (MDR) plug-in is a prototype (as yet unreleased) plug-in aimed at providing a more structured report over the analyses that have been done by Frama-C than a list of ACSL annotations with their status. It features a textual output in the markdown format, that can then easily be translated into a proper document (docx, pdf, html, …). The plug-in can also take additional user remarks (also written in markdown), for instance to explain how it has been checked that an alarm reported by EVA is a false alarm due to the over-approximations made by the tool and not a real issue in the program under analysis. MDR currently focuses on analyses done with the EVA plug-in but could be extended to other kinds of analyses depending on users’ needs.

4.13 RPP plugin

RPP is an experimental plugin that aims at proving Relational Properties (hence the name RPP, standing for Relational Properties Prover) in which several functions calls are involved, as described in the paper “Static and Dynamic Verification of Relational Properties on Self-Composed C Code” [28]. Broadly speaking, a standard ACSL contract (formally) explains what is supposed to occur during a single call of a single function. This is not always the most convenient way to specify what a function should do. For instance, we can consider a function `compare` with the following prototype:

```
int compare(const struct S s1, const struct S s2);
```

Usually, such a function is supposed to implement an ordering relation, meaning in particular that for any structures `s1`, `s2` and `s3`, whenever `compare(s1,s2)` and `compare(s2,s3)` are less than or equal to `0` (meaning that `s1` (resp. `s2`) is less than `s2` (resp. `s3`) according to `compare`), then a call `compare(s1,s3)` must also return a non-positive value (i.e. `compare` must be transitive). Such a specification is quite difficult to encode in pure ACSL, and it would be even more difficult to prove it with the standard WP plugin. RPP provides a concise way to write the specification, and translates that to a set of ACSL clauses over an instrumented version of the code, whose proof by WP implies that the RPP property holds over the original code. A possible RPP specification (assuming `compare` is pure, i.e. does not modify its arguments) would be the following:

```
/*@
requires \valid_read(s1);
requires \valid_read(s2);
assigns \result \from s1, s2;
*/
int compare(const struct S s1, const struct S s2);

/*@ relational transitivity:
  \forall struct S s1, s2, s3;
  \callpure(compare,s1,s2) <= 0 ==>
  \callpure(compare,s2,s3) <= 0 ==>
  \callpure(compare,s1,s3) <= 0;
*/
```

Even though RPP is currently at a very early stage of development, it has been used over a certain number of code snippets and it supports a large subset of C. Within VESSEDIA, it is an integral part of the toolchain that is developed in Task 3.1 for generating lower-level (code) annotations from higher-level (sequence diagrams) models, in order to verify an implementation against said model (see deliverable D3.1 for a complete overview of this toolchain).

From a security properties point of view, RPP is also interesting in itself, as non-interference properties can very easily be expressed as relational properties. Informally speaking, a non-interference property indicates that if a function is called twice over states that only differ on their private parts, the corresponding final states, when the function returns, should have the same public part (i.e. no information over private, sensitive, variables can be inferred by reading the content of public variables). A very simple example of non-interference property expressed in the context of RPP is given below:

```
int secret_variable;
int public_variable;

/*@ assigns \nothing; */
extern void log_access_error(const char* msg);

/*@
  requires \valid(dst) && \valid_read(src);
  assigns *dst \from *src;
*/
void safe_ptr_assign(int *dst, int *src) {
  if (dst == &public_variable && src == &secret_variable)
    log_access_error("Attempt to write secret content in public location");
  else
    *dst = *src;
}

/*@ relational
     non_interference:
      \forall int *dst1, *dst2, *src1, *src2;
        \callset(
          \call(safe_ptr_assign,dst1, src1,id1),
          \call(safe_ptr_assign,dst2, src2, id2))
        ==>
        \at(public_variable, Pre_id1) == \at(public_variable, Pre_id2) ==>
        \at(public_variable, Post_id1) == \at(public_variable, Post_id2);
*/
```

4.14 E-ACSL plugin

Frama-C's E-ACSL plug-in automatically translates an annotated C program into another program that fails at runtime if an annotation is violated. If no annotation is violated, the behaviour of the new program is exactly the same as the one of the original program.

Annotations must be written in the E-ACSL specification language, which is a subset of ACSL. This plug-in is still in a preliminary state: some parts of E-ACSL are not yet supported.

4.15 Metrics collection plugin

Frama-C gathers several static analysis plug-ins in a single collaborative framework. It aims at being correct, that is, never to remain silent for a location in the source code where an error can happen at runtime. As a consequence, if there are no bugs in a program, static analysis is able to detect and to guarantee that property. Frama-C is organized in a plug-in architecture (comparable to that of

Gimp or Eclipse). A common kernel centralizes information and conducts the analyses. Plug-ins interact with each other through interfaces defined by the kernel.

In the scope of VESSEDIA, existing plugins, e.g. ones developed in STANCE project, and several new plugins listed above will provide metric values that will be used in verification of code. The work package WP4 collects and tests the use of various metrics in security evaluation. For this purpose, automated collection of the relevant metrics values provides higher confidence and repeatability of analysis.

The metrics collection plugin will use metric values output from other plugins as input, and will generate a collection of the values.

Data collection from the following plugins and relevant metrics values are planned. This list might change during development, since the plugins themselves that provide the metrics are in development as well. Final list of collected metric values will be described in later deliverables, e.g. in D4.6 [7].

Plugin	Metric	Notes
Frama-C WP	<To be described>	Will be collected via Metrics plugin
EVA plugin	Reachability	numeric value
	Static Analysis Coverage Metric	Ratio
	Quantitative assessment for deductive verification tools	numeric value
	CWE scoring of an alarm	numeric value
	Criticality of an alarm	numeric value
	Liveness metrics	numeric value
	Cryptographic Secrets Persistence	numeric value
	Dangling pointers presistence	numeric value
	Size definition distance	numeric value
Metrics plugin	Collection of existing metrics	Set of numeric values
Report plugin	<To be described>	This plugin is expected to generate formatted output out of metrics collected by the Metrics plugin, which will be processed from the report
SecuRate	Spread of the vulnerability	numeric value
Studia	Criticality of an alarm	boolean
Diversity plugin	Coverage Metric	ratio

Table 10: Metrics: collected by the Metrics collection plugin

Chapter 5 STANCE tools in the security evaluation

The tools described in this chapter were developed in former STANCE project, and are found to be relevant in the VESSEDIA project as well. They are described here as tools forming the basis of development for the VESSEDIA tools described in Chapter 4.

5.1 E-ACSL

This tool is present among the VESSEDIA tools, see above in chapter 0.

5.2 Flinder and FlinderSCA

Flinder fuzzing framework was developed and used in STANCE project by SLAB – see also [17]. In VESSEDIA Project, a similar fuzzing framework integration is planned, but the fuzzing engine will be replaced by AFL – see Chapter 4.2

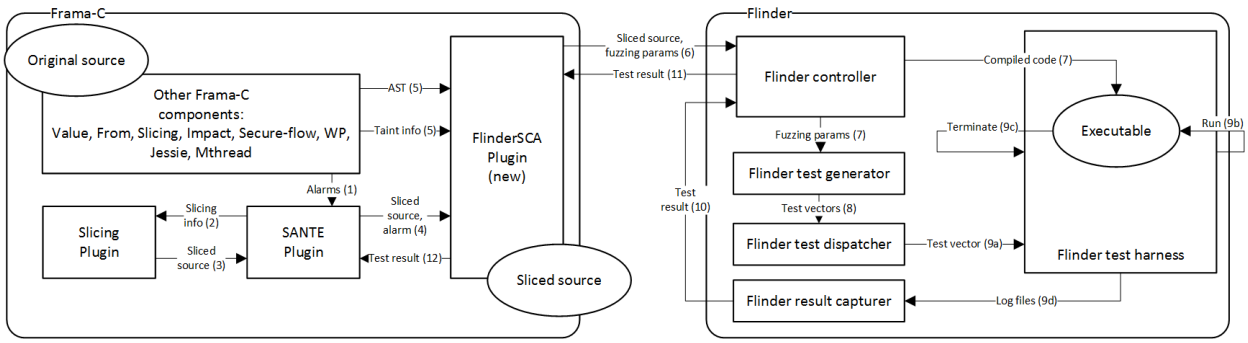

Figure 7: Flinder-SCA Plugin used in FRAMA-C to generate input for Flinder Tool

5.3 PathCrawler

PathCrawler's principal functionality is to automate structural unit testing by generating test inputs which cover all the feasible execution paths of the C function under test.

It can also be used to satisfy other coverage criteria (like k-path coverage restricting the all-path criterion to paths with at most k consecutive loop iterations, branch coverage, MC-DC,...), to generate supplementary tests to improve coverage of an existing functional test suite or to generate just the tests necessary to cover the part of the code which has been impacted by a modification.

PathCrawler can be used to ensure, and demonstrate, code coverage when this is imposed by a standard. However, it can also be used even when code coverage is not imposed, as a convenient and rigorous way of debugging code fragments during development.

Apart from generating tests to ensure coverage, PathCrawler can be used to detect all run-time errors, anomalies such as uninitialized variables or integer overflows and unreachable code.

Another use is to cross-check one implementation against another (previous version or implementation for another platform) or to check conformity with a specification coded in C. PathCrawler will either find test-cases to demonstrate any differences between the results of the two codes or else demonstrate that no such differences exist.

The path tests generated by PathCrawler can also be used to measure the effective execution time of an uninterrupted task in a real-time application, and get an accurate estimate of the longest execution time.

5.4 Frama-C Value

Value has been the first plugin developed for the Frama-C platform in 2005 that implements Abstract Interpretation for the automatic analysis of C source code. It is obsolete and replaced by the EVA plugin (see above). Value computes a value or a set of possible values for each variable in a program. These values belong to integer interval domains. Similar to EVA, many other plugins depend on its results.

5.5 VeriFast

VeriFast is a program verification tool for verifying certain correctness properties of single-threaded and multithreaded C programs. The tool reads a C program consisting of one or more .c source code files (plus any .h header files referenced from these .c files) and reports either “0 errors found” or indicates the location of a potential error. If the tool reports “0 errors found”, this means that the program:

- does not perform illegal memory accesses, such as reading or writing a struct instance field after the struct instance has been freed, or reading or writing beyond the end of an array (known as a buffer overflow, the most common cause of security vulnerabilities in operating systems and internet services) and
- does not include a certain type of concurrency errors known as data races, i.e. unsynchronized conflicting accesses of the same field by multiple threads. Accesses are considered conflicting if at least one of them is a write access. And
- complies with function preconditions and postconditions specified by the programmer in the form of special comments (known as annotations) in the source code.

To detect all errors, VeriFast performs modular symbolic execution of the program.

In STANCE, VeriFast was used to verify certain parts of the reference implementation of the Java API for Trusted Platform Module access. Also, a brief evaluation was made on the feasibility of using VeriFast for verifying a Linux device driver for a Trusted Platform Module.

In VESSEDIA, VeriFast is being extended to better support the verification of cryptographic protocol implementations and the verification of behavioural (I/O) properties of programs. Furthermore, it is being applied to certain parts of the IoT (6LoWPAN) use case, in particular the gateway.

5.6 Frama-C WP

WP (Weakest Precondition) –verifies properties in a deductive manner. Unlike Jessie, it focuses on parameterization with regards to the memory model. WP is designed to cooperate with other Frama-C plugins such as the value analysis plug-in, unlike Jessie that compiles the C program directly into the Why language. WP can optionally use the Why3 platform to invoke many other automated and interactive provers.

This functionality is replicated in the current WP plugin, see Chapter 4.9.

Chapter 6 Summary and Conclusion

The current deliverable described the work carried out towards the Objectives for WP4 set forth in DoA, namely to review existing security evaluation methodologies in the relevant domains, and to investigate how they can benefit from the integration of methods and tools developed during the VESSEDIA project. Relationship to the Common Criteria security standard (CC) was also discussed.

The document proposed security evaluation methodology for the VESSEDIA project, which will be executed in later phases of the project, and will be documented in following deliverables of the WP4 work package. The Security Certification Level (SCL) is also defined, and the security evaluation to be executed is designed to provide certification according to this scheme.

This document also collects and describes the VESSEDIA and STANCE tools developed by partners that will be used in the security evaluation later. The execution of the methodology will also provide results regarding the use of VESSEDIA tools in the security evaluations.

Chapter 7 List of Abbreviations

Abbreviation	Translation
ACSL	ANSI/ISO C Specification Language
CC	Common Criteria
CFI	Code Flow Integrity
CSPN	First Level Security Certification (CERTIFICATION DE SÉCURITÉ DE PREMIER NIVEAU)
DFI	Data Flow Integrity (See 4.3)
EAL	Evaluation Assurance Level
EVA plugin	Evolved Value Analysis plugin (see 4.8)
FAM	Formal Analysis Models (See 4.4)
gcc	GNU Compiler Collection
ICB	Internet Connected Box
IoT	Internet of Things
MDR plugin	MarkDown Report plugin (see 4.12)
RPP plugin	Automatic Proof of Relational Properties by Self-composition plugin (See 4.13)
SCL	Security Certification level (see 2.5)
ToE	Target of Evaluation
VC	Verification Condition
WP	Work Package
WP plugin	Weakest Preconditon plugin (see 4.9)

Table 11: List of Abbreviations

Chapter 8 Bibliography

[1] VESSEDIA DS-01-731453 / D1.1 / V1.0 report: Security requirements for connected medium security-critical applications

[2] VESSEDIA DS-01-731453 / D1.2 / V1.0 report: Requirements descriptions for the WP5 use-cases

[3] VESSEDIA DS-01-731453 / D4.1 report: Metrics for VESSEDIA tools in quality assurance

[4] VESSEDIA DS-01-731453 / D4.3 report (unreleased): Benchmark for evaluating VESSEDIA tools

[5] VESSEDIA DS-01-731453 / D4.4 report (unreleased): VESSEDIA in Common Criteria evaluations

[6] VESSEDIA DS-01-731453 / D4.5 report (unreleased): Quality tests & limits of VESSEDIA tools regarding security vulnerabilities detection

[7] VESSEDIA DS-01-731453 / D4.6 report (unreleased): Evaluation using the VESSEDIA use cases

[8] How to Obtain Common Criteria Certification of Smart TV for Home IoT Security and Reliability by Sooyoung Kang and Seungjoo Kim in Symmetry 2017, 9, 233; doi:10.3390/sym9100233

[9] Jeges E., Berkes B., Eberhardt G. and Kiss B. (2014). MEFORMA Security Evaluation Methodology - A Case Study. In Proceedings of the 4th International Conference on Pervasive and Embedded Computing and Communication Systems - Volume 1: MeSeCCS, (PECCS 2014) ISBN 978-989-758-000-0, pages 267-274. DOI: 10.5220/0004919902670274

[10] Regulation (EU) No 526/2013 of the European Parliament and of the Council of 21 May 2013 concerning the European Union Agency for Network and Information Security (ENISA) - https://publications.europa.eu/en/publication-detail/-/publication/a227aef3-d802-11e2-bfa7-01aa75ed71a1/language-en

[11] ENISA Common Position On Cybersecurity (Dec. 2016) -https://www.enisa.europa.eu/publications/enisa-position-papers-and-opinions/infineon-nxp-st-enisa-position-on-cybersecurity

[12] European Cybersecurity Certification Framework https://ec.europa.eu/digital-single-market/en/eu-cybersecurity-certification-framework

[13] Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on ENISA, the "EU Cybersecurity Agency", and repealing Regulation (EU) 526/2013, and on Information and Communication Technology cybersecurity certification ("Cybersecurity Act"), COM(2017) 477 final/3, Brussels, 1 March 2018 https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CONSIL:ST_12183_2017_REV_2&from=EN

[14] IoTSF Foundation, IoT Security Compliance Framework Release 1.1, December 2017 - https://www.iotsecurityfoundation.org/best-practice-guidelines/#IoTSecurityComplianceFramework

[15] I. Tondel, M. Jaatun, P. Meland, Security Requirements for the Rest of Us: A Survey. IEEE Software, vol. 25, no. 1, pp. 20-27, January/February, 2008

[16] President's Information Technology Advisory Committee, Cyber Security: A Crisis of Prioritization: Report to the President. National Coordination Office for Information Technology Research and Development, 2005

[17] STANCE – FP7 317753 /D4.2.2 C++ code security analysis plug-ins http://www.stance-project.eu/media/Public_Deliverables/STANCE_CEA_DEL_D4.2.2_R1.0.pdf

[18] STANCE – FP7 317753 /D6.1.1 Report on the STANCE security assessment approach for Security evaluation - http://www.stance-project.eu/media/Public_Deliverables/STANCE_SLAB_DEL_D6.1.1_v1.0.pdf

[19] STANCE – FP7 317753 /D6.1.2 Evaluation using the industrial use cases - http://www.stance-project.eu/media/Public_Deliverables/STANCE_SLAB_DEL_D6.1.2_R1.0.pdf

[20] ENISA Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures, November 2017 https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot

[21] ECSO European Cyber Security Certification: A Meta-Scheme Approach, DECEMBER 2017 - http://www.ecs-org.eu/documents/uploads/european-cyber-security-certification-a-meta-scheme-approach.pdf
[22] ANSSI Certification CSPN (French only) - https://www.ssi.gouv.fr/administration/produits-certifies/cspn/
[23] FIRST LEVEL SECURITY CERTIFICATION FOR INFORMATION TECHNOLOGY PRODUCTS (Courtesy Translation) - https://www.ssi.gouv.fr/uploads/2015/01/first_level_security_certification_for_information_en.pdf
[24] EVALUATION CRITERIA FOR THE FIRST LEVEL SECURITY CERTIFICATION (ANSSI-CSPN-CER-I-02/1.1; Courtesy Translation) - https://www.ssi.gouv.fr/uploads/2015/01/evaluation_criteria_for_the_first_level_security_certification_en.pdf
[25] ANSSI Security VISA – CATALOGUE - https://www.ssi.gouv.fr/en/security-visa/security-visa-catalogue/ and https://www.ssi.gouv.fr/uploads/2018/01/catalogue_qualified_solutions_anssi.pdf
[26] Beckert, Bruns, and Grebing, *Mind the Gap: Formal Verification and the Common Criteria* (Discussion Paper). M. Aderhold, S. Autexier, H. Mantel (eds.), VERIFY-2010 (EPiC Series, vol. 3), pp. 4-12.
[27] VESSEDIA H2020 PROJECT: CFI and DFI for runtime protection - Irene Díez-Franco, DeustoTech, University of Deusto (FD); VESSEDIA Technical Meeting 31st May – 1st June 2017 Paris/FRANCE https://vessedia.technikon.com/01-Meetings-Telcos/2017/20170531-0601-Technical-Meeting-Paris/Slides/Day2/05-VESSEDIA-CFI-DFI-runtime-protection-FD.pdf
[28] Static and Dynamic Verification of Relational Properties on Self-Composed C Code - Lionel Blatter, Nikolai Kosmatov, Pascale Le Gall, Virgile Prevosto, Guillaume Petiot (v2 8 May 2018); arXiv:1801.06876 [cs.SE] https://arxiv.org/abs/1801.06876