TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Urve Aavik 242763

Principles and Application of Secure Development in the Public Sector

Master's thesis

Supervisor: Toomas Lepik

Master of science

Tallinn 2026

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Urve Aavik 242763

Turvalise arenduse põhimõtted ja rakendus avalikus sektoris

Magistritöö

Juhendaja: Toomas Lepik

Magister

Tallinn 2026

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Urve Aavik

03.06.2026

Abstract

In recent years, secure development of web applications and APIs has become increasingly important in the public sector due to the growing number and impact of cyber incidents[1]. Public institutions in Estonia rely on (CFR)[2] defined by Estonian Information System Authority and institutional Non-Functional Requirements (MFN)[3] to guide secure software development practices for public-facing digital services. However, it remains unclear whether these frameworks provide sufficient and verifiable security coverage for web applications and APIs when compared to the OWASP ASVS) Level 2[4].

This thesis examines the alignment between CFR, MFN and ASVS Level 2 using an embedded case study in one public-sector organization. A deductive standards-based mapping was performed to compare CFR and MFN requirements with ASVS Level 2 controls, followed by an empirical security assessment of one public-facing information system.

The results show that both CFR and MFN provide only partial alignment with ASVS Level 2, with significant gaps in domains such as authentication, authorization, business logic security, API protection, configuration, and logging. The system-level assessment confirmed weaknesses in the same areas, indicating that requirement-level gaps may lead to practical security risks. The study concludes that the current secure development governance model lacks an explicit verification layer comparable to ASVS Level 2. Integrating a control-level verification standard into development and acceptance processes would improve the reliability and measurability of application security assurance.

This thesis is written in English and is 79 pages long including 4 chapters, 7 figures and 4 tables.

[1] https://www.ria.ee/kuberturvalisuse-aastaraamat-2025
[2] e-gov / cfr · GitLab
[3] MFN.md · main · TRAM-public / MFN · GitLab
[4] GitHub - OWASP/ASVS: Application Security Verification Standard

Lühikokkuvõte

Turvalise arenduse põhimõtted ja rakendus avalikus sektoris

Viimastel aastatel veebirakenduste ja API-de turvaline arendamine muutunud avalikus sektoris üha olulisemak, kuna küberintsidentide arv ja mõju on kasvanud. Eestis tuginevad avaliku sektori asutused tarkvaraarenduse suunamisel Riigi Infosüsteemi Ameti kehtestatud ristfunktsionaalsetele nõuetele (CFR) ning asutusesisestele mittefunktsionaalsetele nõuetele (MFN). Samas ei ole selge, kas need raamistikud tagavad piisava ja kontrollitava turbekatvuse veebipõhistele tarkvaralahendustele võrreldes OWASP Application Security Verification Standardi (ASVS) 2. tasemega.

Käesolevas töös uuritakse CFR-i, MFN-i ja ASVS 2. taseme vastavust ühe avaliku sektori organisatsiooni näitel, kasutades pesastatud juhtumiuuringu meetodit. Töö käigus viidi läbi deduktiivne standardipõhine kaardistus, milles võrreldi CFR-i ja MFN-i nõudeid ASVS 2. taseme kontrollidega, ning sellele järgnes ühe avaliku teenuse infosüsteemi empiiriline turvahindamine.

Tulemused näitavad, et nii CFR kui ka MFN vastavad ASVS 2. taseme nõuetele vaid osaliselt ning märkimisväärsed puudujäägid esinevad eelkõige autentimise, autoriseerimise, äriloogika turbe, API kaitse, konfiguratsiooni ning logimise ja monitooringu valdkondades. Süsteemitaseme hindamine kinnitas samades valdkondades esinevaid nõrkusi, mis viitab sellele, et nõuete tasemel esinevad lüngad võivad väljenduda praktiliste turvariskidena. Töö järeldab, et praeguses turvalise arenduse juhtimismudelis puudub selgesõnaliselt määratletud verifitseerimiskiht, mis oleks võrreldav ASVS 2. tasemega. Kontrollipõhise verifitseerimisstandardi integreerimine arendus- ja vastuvõtuprotsessidesse võimaldaks suurendada veebirakenduste turbe mõõdetavust ja usaldusväärsust.

Lõputöö on kirjutatud inglise keeles ning koosneb 79 leheküljest sisaldades 4 peatükki, 7 joonist ja 4 tabelit.

List of abbreviations and terms

ASVS	Application Security Verification Standard (OWASP application security verification standard)
API	Application Programming Interface
BSIMM	Building Security in Maturity Model
CFR	Cross-Functional Requirements (RIA cross-functional requirements)
CI/CD	Continuous Integration / Continuous Delivery
DAST	Dynamic Application Security Testing
DevSecOps	Development- Security- Operations
IaC	Infrastructure as Code
MFN	Non-Functional Requirements (at the institutional level)
OWASP	Open Worldwide Application Security Project
Oauth	Open Authorization
OIDC	OpenID Connect
PKCE	Proof Key for Code Exchange
RIA	Estonian Information System Authority
SAMM	Software Assurance Maturity Model
SAST	Static Application Security Testing
SBOM	Software Bill of Materials
SCA	Software Composition Analysis
SDL	Security Development Lifecycle (Microsoft)

Table of contents

List of figures

List of tables

Introduction

The importance of Secure Software Development in the public sector has increased significantly in recent years, as both the frequency and impact of cyber incidents have continued to rise (ENISA, 2025) [1]. The security of public-facing web applications and API-based information systems depends not only on the quality of defined requirements but also on their consistent enforcement throughout the entire development lifecycle. In Estonia, the corresponding frameworks are established through the RIA Cross-Functional Requirements (CFR) and institutional Non-Functional Requirements (MFN), which in turn must align with international standards such as the OWASP Application Security Verification Standard (ASVS) Level 2.

In this thesis, I examine the extent to which the Estonian public sector Cross-Functional Requirements (CFR) and institutional Non-Functional Requirements (MFN) align with the OWASP Application Security Verification Standard (ASVS) Level 2. The main objective of this research is to determine whether the existing requirement frameworks provide sufficient and verifiable coverage of ASVS Level 2 controls.

I conduct a structured mapping of CFR and MFN against ASVS Level 2 to assess coverage, identify security gaps and evaluate the risks associated with partially implemented or missing controls. In addition to support the study case I analyze a real-world public sector information system from the ASVS Level 2 perspective to evaluate the practical implementation of these controls.

The thesis is conducted as a qualitative single-case study within one Estonian public sector organization. The research integrates requirement mapping and technical artifact analysis to evaluate both documented alignment and practical enforcement.

The first chapter presents a literature review of organizational security governance frameworks, secure development frameworks and application security verification standards relevant to public-facing web applications and API-based services in the public sector. The review critically compares frameworks such as ISO/IEC 27001, E-ITS and NIST with secure development approaches including SDL, SAMM and DevSecOps,

while positioning OWASP ASVS Level 2 as a granular application-level verification framework for web applications and APIs. The chapter also highlights the conceptual distinction between organizational governance, development process maturity and application-level behavioral verification.

The second chapter, the case study methodology is used to audit the CFR, MFN and real-world public-facing web application and API-based information system using the ASVS document. Also, the improvements for the existing requirement frameworks are made and suggested.

The third chapter presents the empirical findings, including coverage analysis, identified risks and automation feasibility.

The final chapter discusses the results, limitations and practical implications for secure software development in the Estonian public sector.

1 Literature review

This literature review examines organizational security governance frameworks, secure software development models and application security verification standards relevant to public-facing web applications and API-based services in the public sector.

The reviewed approaches represent different abstraction levels within secure software governance. Organizational frameworks such as ISO/IEC 27001 [11], E-ITS [44] and NIST [45] focus primarily on information security management, governance, and risk management at organizational level. Secure development frameworks such as SDL [2], SAMM [47] and DevSecOps [8] address the integration of security practices into the software development lifecycle. In contrast, OWASP ASVS [7] provides application-level verification requirements specifically designed for web applications and APIs.

The purpose of this review is to establish a structured theoretical foundation for evaluating how governance requirements, development practices and application-level verification mechanisms collectively contribute to measurable and verifiable software security assurance.

1.1 Organizational Security and Governance Frameworks

1.1.1 ISO/IEC 27001

ISO/IEC 27001 [11] is an international standard for information security management systems (ISMS). The standard defines organizational requirements for establishing, implementing, maintaining and continuously improving information security management processes.

The framework introduces a risk-based approach for protecting organizational information assets and includes control domains related to access management, supplier relationships, incident management, operational security and business continuity. The controls defined in Annex A provide a baseline for organizational information security governance.

ISO/IEC 27001 is widely adopted in both private and public-sector environments, because it provides a structured governance framework for managing information security risks and demonstrating compliance.

However, ISO/IEC 27001 [43] primarily addresses organizational governance and management processes rather than detailed application-level verification. Although the framework includes software security and secure development related controls, it does not define granular behavioral security requirements for web applications and APIs comparable to OWASP ASVS Level 2. As a result, ISO/IEC 27001 alone cannot provide sufficient assurance regarding the secure implementation of application-specific security controls.

1.1.2 E-ITS

The Estonian Information Security Standard (E-ITS) [44] is the baseline information security framework used in the Estonian public sector. E-ITS is derived from international standards and security practices, including ISO/IEC 27001 and provides a structured catalogue of organizational, technical and procedural security measures.

The purpose of E-ITS is to support public-sector organizations in implementing consistent information security governance, risk management and operational security practices. The framework addresses domains such as asset management, identity and access management, operational continuity, network security, supplier management and incident handling.

E-ITS is important in the context of public-sector software development because it establishes mandatory baseline security requirements for governmental organizations and information systems.

However, like ISO/IEC 27001, E-ITS primarily operates at governance and organizational control level. While the framework includes secure development and software-related measures, it does not define detailed and testable application-level verification requirements for web applications and APIs. Therefore, E-ITS cannot independently function as a comprehensive application security verification standard comparable to OWASP ASVS Level 2.

1.1.3 NIST Cybersecurity Framework and NIST SP 800-53

The NIST Cybersecurity Framework (CSF) [45] defines a high-level risk management model structured around five core functions: Identify, Protect, Detect, Respond and Recover. The framework supports organizations in assessing cybersecurity maturity and prioritizing security activities.

NIST Special Publication 800-53 [46] provides a detailed catalogue of security and privacy controls applicable to information systems and organizations. The controls cover areas such as access control, system integrity, audit logging, configuration management and incident response.

Although NIST frameworks contain controls related to application security [47], they are broader in scope and less focused on granular application-level verification than OWASP ASVS. As a result, they provide strong governance and system-security guidance but limited direct support for validating application behavior under realistic attack conditions.

1.2 Secure Development Frameworks

1.2.1 Microsoft Security Development Lifecycle (SDL)

The Microsoft Security Development Lifecycle (SDL) [2] is one of the earliest structured approaches for integrating security into the software development lifecycle. SDL introduces security-related activities across planning, design, implementation, testing and maintenance phases with the objective of reducing vulnerabilities proactively during development rather than reacting to incidents after deployment.

The framework (Figure 1) emphasizes practices such as threat modelling, secure coding, static code analysis, security testing and vulnerability management. Because of its process-oriented structure, SDL has been widely adopted as a governance model for integrating security into software engineering workflows and making development practices more measurable and repeatable [31].

However, SDL primarily defines development activities and organizational processes rather than granular application-level verification requirements. While the framework described in Figure 1 improves the maturity and consistency of secure development

practices, it does not provide detailed behavioral security controls for validating web application and API security comparable to OWASP ASVS Level 2.

This limitation is particularly relevant in the context of public-facing web services, where secure development processes alone do not guarantee secure runtime behavior. Diao (2025) [4] further argues that traditional lifecycle-based models such as SDL and NIST SSDF are often linear and process-centric, focusing more on procedural integration than on continuous behavioral verification under realistic attack conditions.

As a result, SDL should be understood primarily as a secure development governance framework (Figure 1) rather than a standalone application security verification standard. In practice, SDL is most effective when complemented by granular control-level verification mechanisms such as OWASP ASVS.

Figure 1 Software development life cycle [3].

1.2.2 OWASP SAMM

The OWASP Software Assurance Maturity Model (SAMM)[5] [47] illustrated on Figure 2, is a maturity framework designed to evaluate and improve an organization's secure software development capabilities. SAMM structures software security practices across four domains: Governance, Design, Implementation and Verification, allowing organizations to assess the maturity of their development processes and identify areas for improvement.

Unlike lifecycle-oriented frameworks such as SDL, SAMM focuses on measuring organizational security capability and process maturity rather than prescribing a fixed development methodology. The framework illustrated on Figure 2, provides measurable maturity levels and supports continuous improvement of secure development practices within software engineering environments [6].

SAMM is particularly valuable for organizations seeking to institutionalize secure development practices and establish repeatable governance processes. However, like other maturity-based frameworks, SAMM primarily evaluates the existence and maturity of security processes rather than verifying the behavioral correctness of web applications and APIs at control level.

This distinction is important in the context of public-facing web services, where mature development processes do not necessarily guarantee secure runtime behavior or resistance against application-level abuse cases. While SAMM can help organizations improve secure development governance, it does not provide the granular and testable verification controls defined in OWASP ASVS Level 2.

Therefore, SAMM should be understood as a development maturity and governance framework that complements, but does not replace, application-level verification standards such as ASVS.

[5] The Model

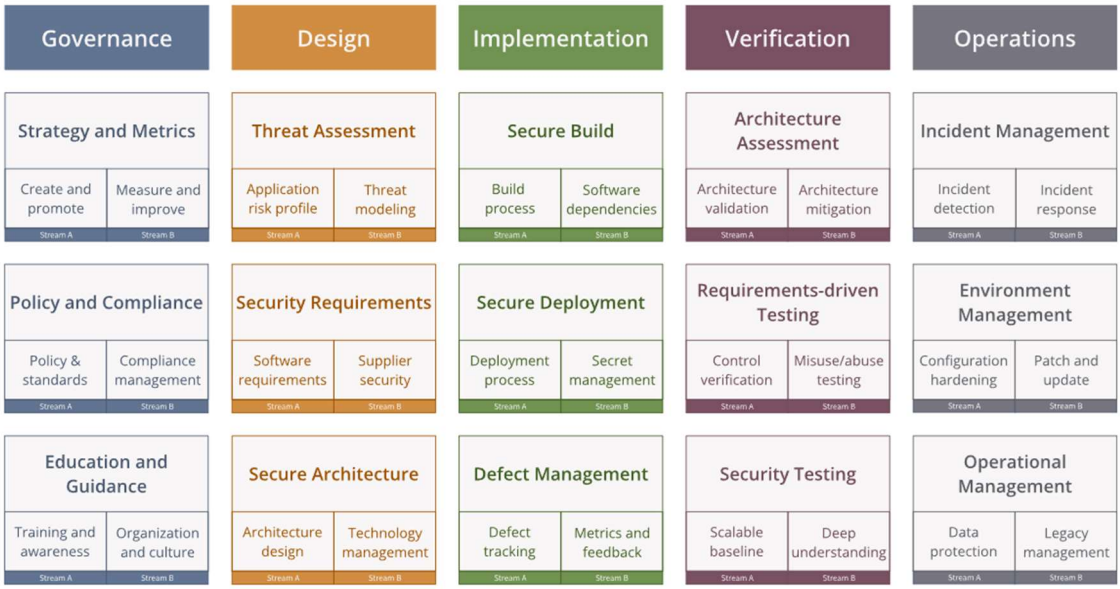


Figure 2 OWASP SAMM Model [5, 47].

1.2.3 DevSecOps and Compliance-as-Code

DevSecOps is a software development approach [8] that integrates security practices directly into automated development and deployment workflows. Unlike traditional security models where security assessment is performed primarily at the end of the development lifecycle, DevSecOps embeds security controls and compliance validation into continuous integration and continuous delivery (CI/CD) pipelines.

In the context of public-facing web applications and API-based services, DevSecOps is particularly important because public-sector organizations increasingly rely on automated deployment pipelines and continuous delivery practices. Studies by Lee and Park (2025) [9] and Smith and Jones (2022) [10] demonstrate that policy-as-code and compliance-as-code mechanisms can significantly improve consistency and reduce human error in security enforcement. Similarly, Rahman and Williams (2021) [13] identify automation and cross-functional collaboration as key factors supporting secure deployment outcomes.

On Figure 3 illustrated DevSecOps model primarily defines an operational and automation-oriented delivery model rather than a standalone application security verification standard. Automated pipelines can verify only those controls that are

explicitly defined and technically measurable. As a result, DevSecOps requires a complementary control framework that specifies what security properties must be validated within the pipeline.

This limitation is particularly important in application-level security verification. While DevSecOps can automate security enforcement and evidence collection, it does not itself define granular behavioral security requirements for web applications and APIs. Therefore, frameworks such as OWASP ASVS Level 2 are necessary to provide the detailed verification criteria that DevSecOps pipelines can implement and validate automatically.

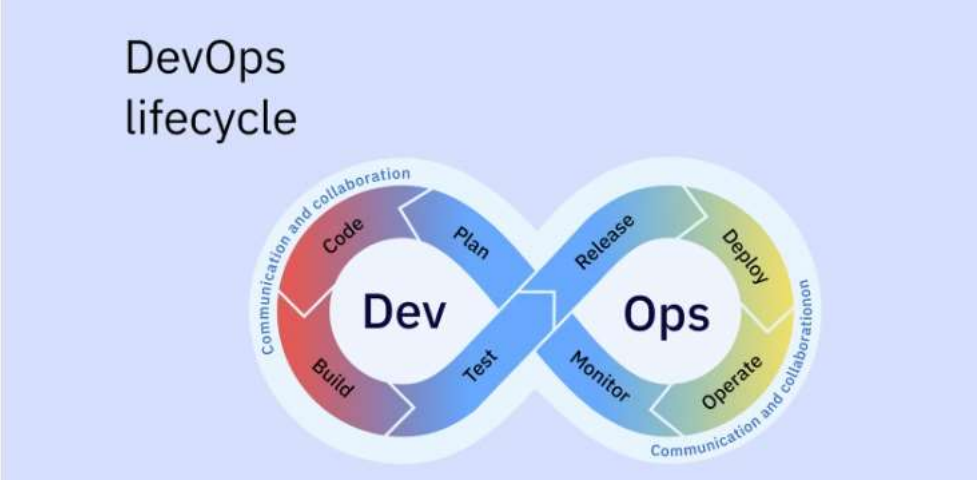


Figure 3 DevSecOs process [8].

1.2.4 SBOM and Supply Chain Visibility

On Figure 4 illustrated Software Bill of Materials (SBOM)[6] is a structured, machine-readable inventory of all components. It is used in a software product, including third-party libraries, dependencies, and their versions [35]. Its purpose is to provide transparency into the software supply chain, enabling organizations to identify vulnerabilities, manage risks and verify compliance more effectively [15].

Xia et al. (2023) [16] and Nocera et al. (2025) [17] emphasize the role of SBOM in tracking third-party components and reducing vulnerabilities, while Papaioannou et al. (2025) [18] and Zhang et al. (2025) [19] highlight the challenges related to SBOM

[6] Software Security in Supply Chains: Software Bill of Materials (SBOM) | NIST

standardization and the diversity of available tooling ecosystems. These studies [30] confirm that SBOM should be an integral artifact within the DevSecOps process, providing verifiable evidence for assessing compliance with ASVS requirements and evaluating the reliability of software components.

On Figure 4 illustrated SBOM primarily improves component visibility and supply-chain transparency rather than validating application behavior or runtime security controls. Although SBOM supports compliance verification and vulnerability management, it does not define application-level security requirements comparable to OWASP ASVS Level 2. Therefore, SBOM should be understood as a complementary assurance mechanism that strengthens software transparency and operational risk management but does not replace control-level application security verification.

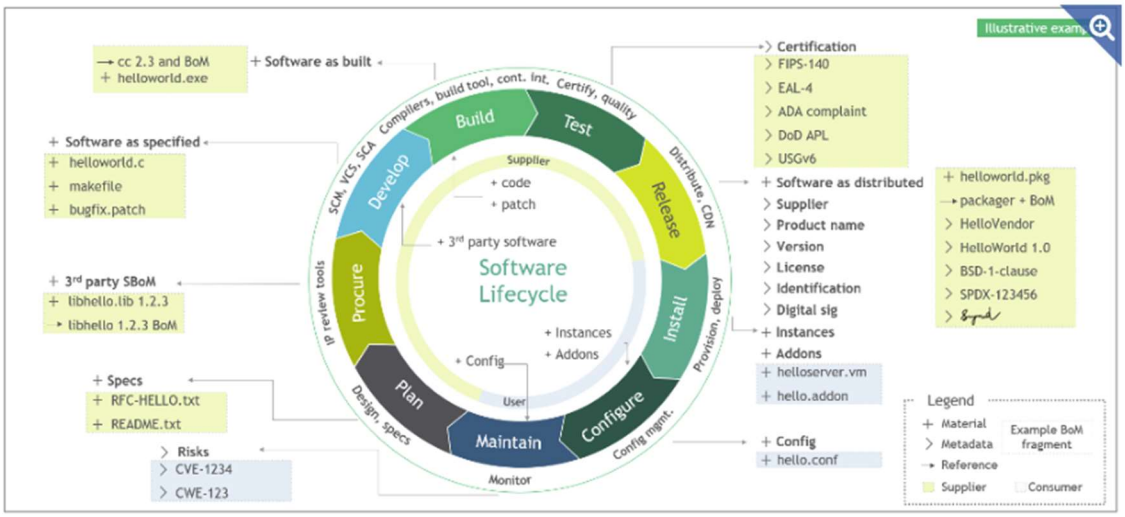


Figure 4 Software Life Cycle and Bill of Materials Assembly Line [15].

1.3 Application Security Verification Standards

1.3.1 OWASP ASVS

The OWASP Application Security Verification Standard (ASVS) [7] is a structured application security verification framework specifically designed for web applications and APIs, as illustrated in Figure 4.

Its purpose is to provide a clear and measurable set of controls that can be used to design, develop and verify secure applications [7]. ASVS organizes requirements into categories such as authentication, access control, validation, cryptography and error handling, making it a practical checklist for both developers and auditors [32].

Figure 4 illustrates the three ASVS verification levels.

Level 1 represents a basic security baseline suitable for low-risk applications. It primarily addresses common vulnerabilities, such as those included in the OWASP Top 10 and is the only level that can typically be validated through penetration testing alone. Level 1 is generally sufficient for applications that do not store or process sensitive data.

Level 2 applies to applications that handle sensitive data or business-critical functionality. It requires more structured security verification, including review of documentation, configuration and development practices. Level 2 introduces broader control coverage across authentication, access control, cryptography, logging and secure configuration. It is intended for most business applications and systems that process protected or regulated information.

Level 3 is reserved for high-risk or mission-critical systems, such as military, healthcare, safety, or critical infrastructure applications. It requires in-depth architectural analysis, detailed documentation, modular security design and comprehensive verification of confidentiality, integrity and availability controls. Level 3 is appropriate where system failure could severely impact organizational operations or public safety.

The latest version of OWASP ASVS is 5.0 [8] published May 2025. ASVS 5.0 differs from version 4.0 [9] primarily in its refined scope and structure: requirements are rewritten to focus on preventing security flaws rather than prescribing specific technical implementations and they are made more self-explanatory and verification oriented. Version 5.0 introduces documented security decision requirements, updated level definitions for easier adoption and a restructured and expanded control set (approximately 350 requirements across 17 chapters) with formal mapping support from v4.0 [33].

[7] OWASP Application Security Verification Standard (ASVS) | OWASP Foundation
[8] ASVS/5.0 at master · OWASP/ASVS · GitHub
[9] ASVS/4.0 at master · OWASP/ASVS · GitHub

Unlike organizational governance frameworks such as ISO/IEC 27001, E-ITS or NIST CSF, ASVS focuses directly on application behavior and control-level verification. Unlike maturity models such as SAMM, which assess organizational development capability, ASVS defines granular and testable technical controls that can be directly verified through code review, configuration analysis and security testing.

This distinction is particularly important in the context of public-facing web applications and API-based services, where secure development governance alone does not guarantee secure application behavior in practice.

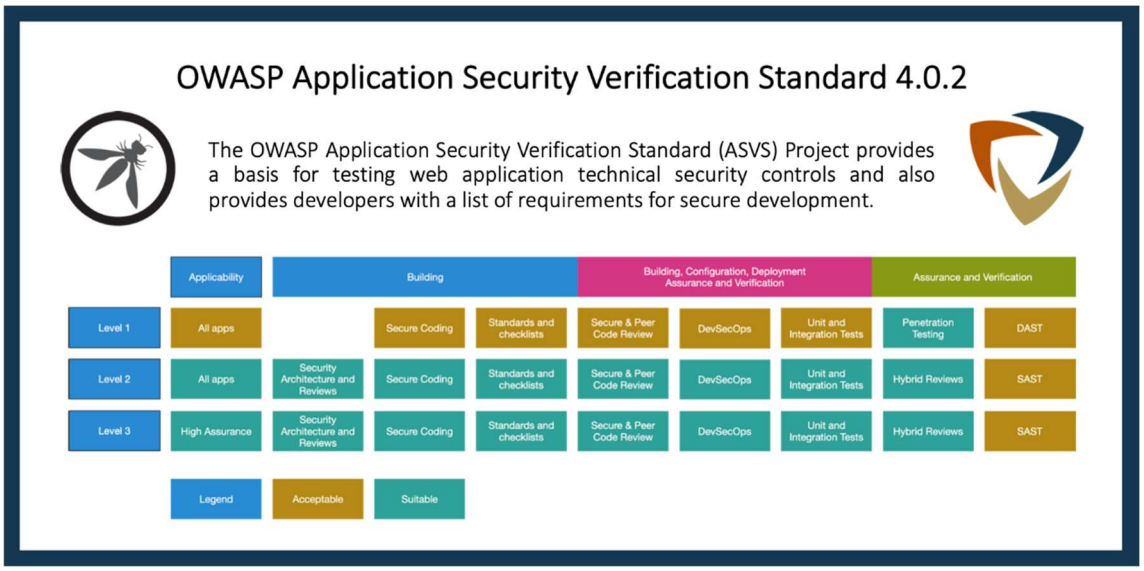


Figure 5 OWASP Application Security Verification Standard 4.0 Levels [7].

1.4 Comparative Positioning of Frameworks

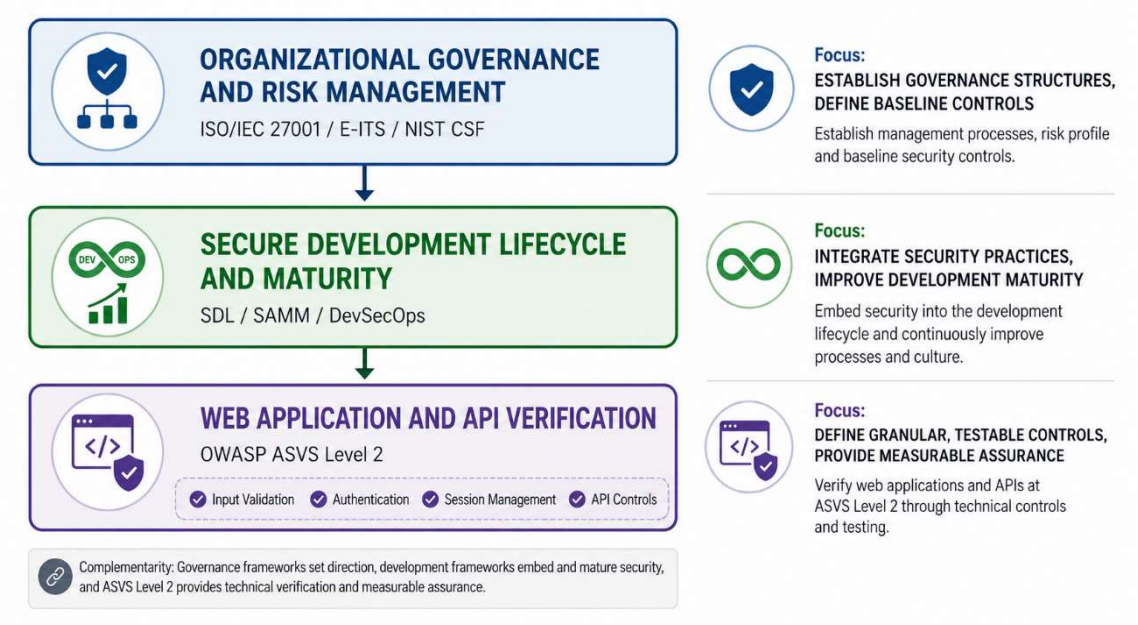


Figure 6 Relationship between organizational governance frameworks, secure development frameworks and OWASP ASVS Level 2 verification layer.

As illustrated in Figure 6[10], the reviewed frameworks operate at different levels of abstraction within secure software governance and therefore address different security objectives..

Organizational governance frameworks such as ISO/IEC 27001, E-ITS and NIST CSF focus primarily on risk management, operational security and organizational compliance. Their purpose is to establish security governance structures and baseline controls across the organization.

Secure development frameworks such as SDL, SAMM and DevSecOps focus on integrating security practices into the software development lifecycle and improving organizational development maturity.

In contrast, OWASP ASVS defines detailed and testable application-level security controls specifically intended for verification of web applications and APIs.

[10] https://chatgpt.com/

This distinction is important because compliance with organizational governance frameworks does not automatically guarantee secure application behavior. Governance-oriented frameworks typically define expected outcomes and management processes, whereas ASVS defines granular behavioral controls that can be directly validated through testing and technical verification. Therefore, the frameworks should be understood as complementary rather than competing approaches.

2 Research method

This thesis uses a qualitative embedded single-case study to examine the alignment between Estonian public-sector software development requirement frameworks (CFR and MFN) and the OWASP Application Security Verification Standard (ASVS) Level 2 within a real institutional context [20].

A case study approach is appropriate when the phenomenon under investigation is closely connected with its organizational and technical environment and cannot be meaningfully analyzed in isolation. In cybersecurity research [37], such conditions are common because development requirements, governance rules and system implementations form a tightly coupled socio-technical context. The research design combines two complementary methodological components. First, a deductive [38] standards-based document analysis is used to compare CFR and MFN requirements against ASVS 5.0 Level 2 controls. Second, an applied descriptive [39] system-level security assessment is used to evaluate one public-facing information system using the same ASVS Level 2 control framework.

ASVS Level 2 is used as a common analytical benchmark to ensure consistency between requirement-level analysis and system-level evaluation.

The study follows the principle of methodological triangulation by combining document analysis, control-level comparison, empirical assessment results, and expert clarification to increase the reliability of the findings [21].

2.1 Methodology

2.1.1 Research strategy

This thesis adopts **an embedded qualitative single-case study** design in the context of public-sector secure software governance for public-facing web applications and API-based services. The study investigates whether the development requirements used in one Estonian public-sector organization provide sufficient and verifiable application-level security assurance when evaluated against the OWASP Application Security Verification Standard (ASVS) 5.0 Level 2.

The research follows an observational approach, as the study does not manipulate variables or create controlled experimental conditions. Instead, the research examines existing development requirements and an operational information system within their real organizational and technical environment. According to Edgar and Manz (2017) [40], observational research is appropriate in cybersecurity studies when the phenomenon cannot be isolated from its operational context and when the goal is to understand how security controls function in practice rather than under laboratory conditions.

The case study is **embedded**, because it contains multiple units of analysis within a single organizational case. The study includes both requirement-level analysis and system-level assessment, which are examined using different but complementary methods. This design allows the research to evaluate both the normative level (documented requirements) and the implementation level (actual system behaviour).

The methodological approach combines two observational methods:

1. **Deductive standards-based document analysis** used to evaluate the coverage of the Cross-Functional Requirements (CFR) [22] and institutional Non-Functional Requirements (MFN) [23] against OWASP ASVS 5.0 Level 2 controls.
2. **Applied descriptive case study** based on a structured white-box security assessment of one public-facing information system.

This combination makes it possible to determine not only whether security requirements are formally defined, but also whether the absence of specific controls can be observed in practice.

2.1.2 Methodological components

The research consists of two complementary methodological components. The first component is a **deductive qualitative content analysis of policy and requirement documents.** In deductive content analysis, the coding categories are defined in advance based on an existing theoretical or normative framework and the analyzed documents are systematically mapped against these predefined categories (Elo & Kyngäs, 2008) [41]. This approach is appropriate when the aim is to test the extent to which existing documentation conforms to a known standard. In this study, OWASP ASVS 5.0 Level 2 is used as the analytical framework, and the requirement sets CFR and MFN are treated

as the analyzed documents. Each ASVS control is evaluated against the requirements and classified as fully covered, partially covered, or not mapped.

Document analysis is used as the primary technique for reviewing written requirements and classifying them according to predefined control categories (Bowen, 2009) [42]. Because the analysis is performed using a predefined control framework, the method can also be described as **standards-based document analysis.**

The second component is an **applied descriptive case study** based on a structured security assessment of one operational information system. According to Edgar and Manz (2017), applied descriptive research is suitable when the objective is to examine how a system, process or control mechanism performs in a real-world environment without experimental manipulation. In this study, the purpose of the system-level assessment is not to prove causality, but to observe whether the types of control gaps identified at requirement level are also visible in practice. The system assessment was conducted as a white-box security evaluation based on ASVS Level 2 control categories and included manual testing, configuration inspection, and abuse-case validation.

2.1.3 Units of analysis and analytical benchmark

The embedded case study contains three units of analysis:

1. the **Cross-Functional Requirements (CFR)** used in public-sector development governance,
2. the **institutional Non-Functional Requirements (MFN)** used at organizational level,
3. one **selected public-facing information system** developed under these requirements.

All three units are evaluated using the same analytical benchmark, which is OWASP Application Security Verification Standard (ASVS) 5.0 Level 2.

ASVS Level 2 was selected because it defines control-level security requirements for web applications and APIs and allows consistent verification of whether specific security

controls are implemented. Using the same benchmark for all units enables direct comparison between documented requirements and observed system behaviour.

2.1.4 Mapping procedure

For the document analysis, each ASVS Level 2 control was compared against the requirement sets CFR and MFN.

Each control was classified into one of three categories:

1. Full – the requirement explicitly defines a verifiable control equivalent to the ASVS control.
2. Partial – the requirement addresses the control conceptually or at architectural level but does not define verifiable implementation criteria.
3. Not mapped – no corresponding requirement is present.

The results were summarized as:

- total number of mapped controls,
- percentage of coverage,
- distribution of unmapped controls by ASVS domain,
- identification of high-risk gaps.

This classification scheme follows the logic of deductive content analysis, where predefined categories are applied to the data (Elo & Kyngäs, 2008).

2.1.5 System-level assessment procedure

The empirical part of the study is based on an independent white-box security assessment of one public-facing information system developed under the analyzed requirements.

The assessment was performed by a technical security tester using a structured methodology aligned with ASVS Level 2 control domains.

The evaluation included:

- manual security testing,
- inspection of application behaviour and server responses,

- verification of security controls against ASVS categories,
- validation of abuse-cases such as injection, access-control bypass and improper input handling,
- evaluation of configuration and deployment settings.

White-box testing conditions included access to application source code and HTTP interface, while direct server, database and log access were not available.

The purpose of the system-level assessment is not to evaluate the quality of a single system, but to determine whether requirement-level gaps identified in the document analysis can also be observed at implementation level.

2.1.6 Expected outputs

The expected outputs of the study are:

- a control-level mapping of CFR against ASVS Level 2,
- a control-level mapping of MFN against ASVS Level 2,
- quantitative coverage results,
- domain-level gap distribution,
- identification of high-risk missing controls,
- empirical observations from system-level audit,
- synthesis of normative and empirical results.

The combination of document analysis and system assessment allows the study to determine whether existing public-sector development requirements provide sufficient and verifiable application-level security.

2.1.7 Validity and limitations

The study is limited to a single organizational case and one selected information system, which restricts statistical generalization. However, the use of a well-defined external benchmark (ASVS Level 2) enables analytical generalization.

The system assessment was performed under partial white-box conditions, without direct access to server configuration, database, or log files, which may limit the ability to verify certain controls.

Despite these limitations, the use of two complementary methods - deductive document analysis and applied descriptive system assessment- provides triangulation between requirement-level and implementation-level evidence, increasing the reliability of the conclusions.

2.1.8 Research questions and method mapping

The research design combines document analysis and system-level security assessment to answer the research questions of the study. Each research question is addressed using a specific method and data source, while all analyses use OWASP ASVS 5.0 Level 2 as the common evaluation benchmark.

RQ1. To what extent do the public-sector Cross-Functional Requirements (CFR) cover OWASP ASVS 5.0 Level 2 controls?

Method: Deductive standards-based document analysis.

Data source: CFR requirement documentation.

Procedure: Each ASVS Level 2 control is compared against CFR requirements and classified as Full, Partial, or Not mapped.

Expected result: control-level mapping matrix CFR → ASVS, percentage of coverage, distribution of missing controls by ASVS domain, identification of high-risk gaps.

Purpose: To evaluate whether national-level development requirements define verifiable application security controls.

RQ2. To what extent do the institutional Non-Functional Requirements (MFN) cover OWASP ASVS 5.0 Level 2 controls?

Method: Deductive standards-based document analysis.

Data source: MFN requirement documentation.

Procedure: Each ASVS Level 2 control is compared against MFN requirements and classified as Full, Partial, or Not mapped.

Expected result: control-level mapping matrix MFN → ASVS, percentage of coverage,

domain-level gap distribution, identification of partially defined and missing controls.

Purpose: To determine whether organizational-level requirements provide more detailed and verifiable security guidance than national-level requirements.

RQ3. Do requirement-level gaps identified in CFR and MFN appear in practice at system level?

Method: Applied descriptive case study based on white-box security assessment.

Data source: independent security assessment report, test results, observed system behavior, configuration responses.

Procedure: Security findings are classified according to ASVS Level 2 domains and compared with the gaps identified in the document analysis.

Expected result: distribution of findings by ASVS domain, identification of missing controls in practice, confirmation or rejection of requirement-level gap hypotheses.

Purpose: To determine whether the absence of controls in requirement documents is observable in the actual implementation of a public-facing system.

RQ4. Do CFR and MFN together provide sufficient and verifiable application-level security coverage comparable to ASVS Level 2?

Method: Cross-case synthesis of document analysis and system-level assessment.

Data source: CFR mapping results, MFN mapping results, system audit results.

Procedure: Results from all analyses are compared to determine overall coverage level,

consistency between requirements and implementation, presence of systematic gaps, adequacy of the requirement model.

Expected result: overall coverage evaluation, identification of structural weaknesses,

methodological conclusion about adequacy of the current requirement framework.

Purpose: to assess whether the current public-sector requirement model provides sufficient, testable, and implementation-level security assurance.

Methodological consistency

All research questions use the same analytical benchmark, OWASP ASVS 5.0 Level 2, which ensures consistency between document analysis and system-level assessment.

The combination of deductive document analysis and applied descriptive system evaluation enables triangulation between normative requirements and observed implementation, increasing the validity of the conclusions.

2.2 Research Subject

The case study is conducted within the Estonian Transport Administration, a public-sector organization that applies Cross-Functional Requirements (CFR) issued by the Estonian Information System Authority (RIA) and institutional Non-Functional Requirements (MFN) in its software development projects. These requirements define security expectations for public-facing information systems developed or maintained under the institution's governance.

The author of this thesis is professionally involved in IT governance processes within the public sector. Such access enables detailed analysis of requirement documentation, technical artifacts and development practices relevant to the enforcement of security controls. The organizational context allows evaluation of both normative requirements (CFR and MFN) and their practical implementation in real development workflows.

The empirical analysis focuses on one selected public-facing information system developed under the Transport Administration's requirement framework. The system is evaluated against OWASP Application Security Verification Standard (ASVS) Level 2 controls.

The case is relevant because public institutions increasingly rely on CI/CD-based development pipelines, where security enforcement must be automated and repeatable. Without structured verification and automation, control implementation may become

inconsistent. This study therefore examines both documented alignment (CFR/MFN → ASVS Level 2) and practical enforcement within an operational environment.

The organizational requirement structure (CFR and MFN) is described in Appendix 1 (Appendix 1 – Cross-Functional Requirements (CFR)) and in Appendix 2 – Institutional Non-Functional Requirements (MFN).

2.2.1 RIA overview

The Estonian Information System Authority (RIA) [25] is the national agency responsible for coordinating and securing public information systems and digital infrastructure. Operating under the Ministry of Justice and Digital Affairs, RIA's mandate includes strengthening national cybersecurity, managing cyber incidents, and supporting secure digital services across government. Its responsibilities also encompass the oversight of critical state IT components such as the X-Road data exchange layer and the national information system registry (RIHA) [26].

RIA issues the Cross-Functional Requirements (CFR) framework, which defines high-level security expectations for public sector information systems and software development projects. CFR sets overarching security expectations that public organizations must translate into verifiable and enforceable mechanisms within their development and delivery environments. This positions CFR as a bridge between national policy objectives and practical secure software development, particularly when evaluated against detailed standards such as OWASP ASVS Level 2.

2.2.1.1 CFR Structure and Components

The Cross-Functional Requirements (CFR) framework defines baseline requirements for public-sector information systems across development, delivery, architecture, security, and data governance. The requirements are organized into thematic categories and assigned obligation levels (mandatory, expected, recommended). CFR establishes principles and expected outcomes rather than prescribing specific technical implementations.

In the development and delivery domains, CFR requires version-controlled source code, maintainable design, licensing transparency, database migration capability, automated testing, security testing prior to release, code review and automated deployment. These

provisions support traceability, controlled release management and DevSecOps-aligned enforcement.

Architecturally, CFR emphasizes centralized identity management, secure communication (TLS), REST-based APIs, microservice-oriented design, X-Road interoperability, cloud readiness and externalized configuration. Security controls include static code analysis, input validation, cryptographic compliance, logging, and resilience against external failures.

In data governance, CFR requires structured data descriptions, standardized formats, machine-readable exports and quality monitoring mechanisms.

Overall, CFR functions as a high-level governance framework. Because it defines principles rather than detailed verification criteria, systematic mapping against OWASP ASVS Level 2 is necessary to assess control coverage and identify security gaps.

2.2.1.2 MFN Structure and Components

The Transport Administration's Institutional Non-Functional Requirements (MFN) are based on the national Cross-Functional Requirements (CFR) issued by RIA but provide a more detailed and implementation-oriented specification. While CFR defines high-level security and governance principles, MFN translates these into concrete technical and operational controls applicable within the institutional development environment.

MFN specifies enforceable requirements for coding standards, testing practices, CI/CD automation, database structure, logging formats, cryptographic controls, deployment procedures, and accessibility compliance. Compared to CFR, the requirements are more prescriptive and measurable, enabling direct verification within development pipelines and operational systems.

This detailed structure makes MFN particularly suitable for systematic mapping against OWASP ASVS Level 2, as many controls can be directly evaluated against observable implementation evidence. The full MFN specification is publicly available in the Transport Administration repository [23].

2.2.1.3 Comparison of CFR and MFN

Although the Cross-Functional Requirements (CFR) and the Institutional Non-Functional Requirements (MFN) are structurally related and partially overlapping, they do not constitute duplicated or equivalent documents. Instead, they fulfil different roles and operate at different normative and operational levels within the public-sector software development governance model, as summarized in Table 1.

Structural Similarities

Both CFR and MFN define requirements applicable to public-sector information systems and aim to ensure secure, maintainable and compliant software development. In several areas, the documents demonstrate direct or near-direct overlap. These include:

- Use of secure communication (e.g., TLS).
- Static code analysis and security testing before production release.
- Input validation and secure coding principles
- Protection of secrets (no hard-coded credentials)
- Cryptographic compliance with national guidelines
- Structured logging and audit trail requirements
- Licensing transparency and source code version control

About one third of the CFR requirements are directly reflected in the MFN in a nearly one-to-one manner. In these cases, MFN turns the high-level principles defined in CFR into more specific and enforceable institutional controls.

A further portion of requirements demonstrate thematic alignment but differ in level of abstraction. For example, CFR may define architectural principles such as cloud readiness, microservice orientation or API-based interoperability. MFN specifies implementation-level controls related to CI/CD automation, configuration management and deployment procedures. These cases represent partial overlap, where both documents address the same domain but at different levels of granularity.

Structural and Functional Differences

Despite partial overlap, approximately one-third of CFR requirements address domains that are not explicitly covered by MFN. These include:

- Architectural design paradigms (e.g., Domain-Driven Design, microservices)
- National interoperability obligations (e.g., X-Road integration)
- Data governance and registry publication requirements (e.g., RIHA documentation)
- Long-term lifecycle and end-of-life (EOL) planning
- Standardization of classifiers and identifiers
- Accessibility compliance
- Broader governance and strategic IT alignment principles

These requirements extend beyond application-level security controls and instead reflect broader governance, interoperability, and quality management considerations.

The key functional distinction between the documents can be summarized as follows:

Table 1 CFR and MFN comparison.

CFR	**MFN**
Governance and architectural principles	Technical and verifiable control requirements
High-level normative expectations	Concrete implementation and enforcement rules
Broad coverage of data governance and interoperability	Focus on application-level security and CI/CD enforcement
Indirect relation to ASVS	Directly mappable to ASVS control structure

Systemic Interpretation

Although CFR and MFN show a similar overall ASVS Level 2 coverage percentage in quantitative analysis, this does not imply substantive equivalence. The similarity in

coverage metrics results from partial thematic overlap rather than duplication of normative content.

CFR functions primarily as **a high-level governance and architectural framework**, defining expected outcomes and strategic alignment requirements across public-sector IT systems. **MFN**, in contrast, operates as **an institutional enforcement mechanism**, translating selected principles into concrete, measurable controls suitable for verification within development pipelines and operational environments.

Therefore, the documents should be understood as complementary rather than redundant. **CFR establishes** the **broader regulatory and architectural context**, while **MFN** operationalizes **security controls within a specific institutional development environment**.

2.2.2 Company overview

The Estonian Transport Administration (Transpordiamet) is a government agency responsible for planning, regulating and supervising transportation systems across land, air and water within Estonia. Established on 1 January 2021 through the merger of the Civil Aviation Administration, Road Administration and Maritime Administration. It serves as a unified competence center for all modes of transport. The agency's mission is to ensure safe, smart and environmentally friendly mobility. [27].

In addition to regulatory and supervisory responsibilities, the agency operates multiple public-facing digital information systems that support citizens, businesses and sector stakeholders. Among the most significant systems are the Traffic Register [28], which manages vehicle and driver data; the Electronic Maritime Information System [29]; the aviation safety information system LOIS and maritime-related registries such as the Seafarers Information System, the Ship Information System, the Port Register. These systems process personal data, operational records and regulatory information, making secure software development and continuous compliance important.

2.2.3 Development Lifecycle

Software development in the Transport Administration is governed by the internal IT Development Projects Procedure ilustrated in Figure 7. The process is designed to ensure that initiatives are aligned with organizational priorities and delivered within

agreed scope, schedule, and budget, supported by required documentation and risk consideration.

Development typically starts from a proposal initiated by a business owner or triggered by legislative changes, operational incidents or security vulnerabilities. Proposals are assessed and depending on cost and impact, reviewed by the Development Proposals Committee. Execution follows a structured flow of analysis, development, testing and controlled production deployment, with formal acceptance documentation required before invoicing.

The Transport Administration does not maintain in-house system analysts, developers or testers- these roles are primarily procured through external development partners. To ensure consistent implementation, MFN is included in procurements and technical specifications as a binding set of requirements, including security, testing, logging and deployment controls.

This lifecycle provides the practical governance context in which CFR-derived MFN requirements are applied in projects. It also explains why mapping CFR and MFN to OWASP ASVS Level 2 is meaningful: ASVS controls can be compared both to documented requirements (CFR/MFN) and to the evidence produced through delivery activities.

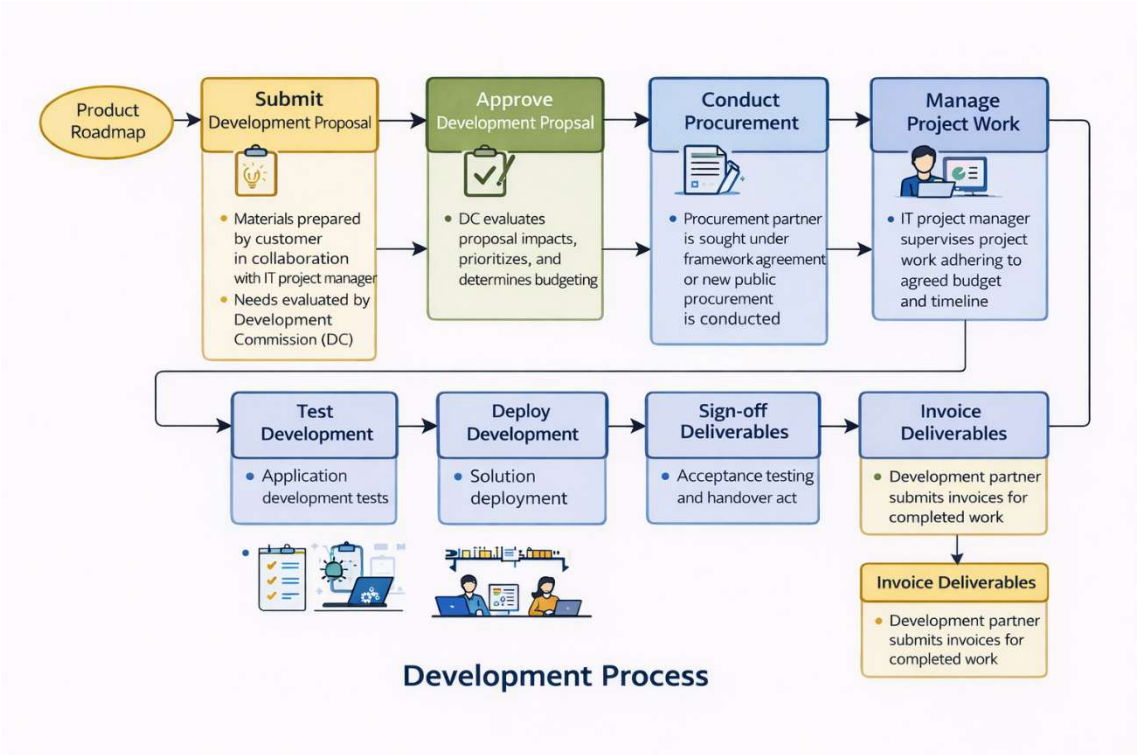


Figure 7 Development Process.

2.2.4 Standard selection

This thesis uses the OWASP Application Security Verification Standard (ASVS) as the primary benchmark for security assessment. ASVS was selected because it is specifically designed for verification of web applications and API-based services, which correspond to the dominant architecture of modern public-sector digital services. ASVS defines control-level, testable security requirements for web applications and APIs, which makes it suitable for systematic comparison between documented requirements and observable implementation behavior. Unlike general security frameworks, ASVS provides detailed verification criteria that allow each control to be evaluated individually. According to OWASP, ASVS is designed to support both secure development and security verification by defining a structured set of security controls that can be used by developers, auditors and security testers to assess the security of an application at different assurance levels.

ASVS defines three assurance levels. Level 1 provides basic protection suitable for low-risk applications, Level 2 defines a standard level of security suitable for most business applications, and Level 3 defines high-assurance requirements for critical systems.

Level 2 was selected for this study because it represents the recommended baseline for applications that process sensitive data or are exposed to untrusted networks. Public-sector information systems typically fall into this category, as they often process personal data, provide public-facing services, or interact with other government systems.

Alternative frameworks such as ISO/IEC 27001, NIST SP 800-series[11], or OWASP SAMM were considered, but they were not used as the primary analytical framework for the following reasons. ISO/IEC 27001 defines requirements for information security management systems at the organizational level, but it does not provide detailed control-level requirements for application security. NIST publications provide extensive guidance on security controls, but they are primarily oriented toward system and infrastructure security rather than application-level verification. OWASP SAMM defines a maturity model for secure development processes, but it evaluates organizational practices rather than individual application security controls. In contrast, ASVS provides a control-level verification standard specifically designed for application security, which makes it suitable for deductive document analysis and system-level assessment.

Nevertheless, frameworks such as SAMM, NIST SSDF and DevSecOps-oriented compliance models remain complementary to ASVS because they address organizational maturity, lifecycle governance and automation capabilities that ASVS itself does not comprehensively define.

2.2.5 MFN audit

2.2.5.1 Objective and Scope

The objective of this analysis is to evaluate the extent to which the Transport Administration's Institutional Non-Functional Requirements (MFN) provide coverage of the OWASP Application Security Verification Standard (ASVS) 5.0 Level 2 controls.

The central research question addressed in this section is whether MFN alone can provide application-level security coverage comparable to ASVS Level 2, and in which domains significant gaps remain. The assessment covers all ASVS 5.0 Level 2 controls across chapters V1–V17, comprising a total of 263 individual security controls.

[11] NIST Special Publication 800-series General Information | NIST

2.2.5.2 Methodology

The mapping procedure is described in paragraph 2.1.4.

2.2.5.3 Quantitative Results

In total, results are following:

- OWASP ASVS 5.0 Level 2 controls (V1–V17): **263**
- Controls mapped in MFN (Full + Partial): **90**
- Controls not mapped in MFN: **173**

This indicates that MFN provides **coverage for approximately 34%** of ASVS 5.0 Level 2 controls, while 66% of controls remain without a direct equivalent in MFN (Table 2).

These figures demonstrate that MFN does not provide comprehensive application-level security coverage when evaluated against ASVS Level 2.

2.2.5.4 Distribution of Unmapped Controls by Domain

Table 2 Distribution of Unmapped OWASP ASVES Level 2 Controls in MFN by Domains.

ASVS Chapter	Domain Name	Unmapped L2 Controls	Share of Total L2 Controls (%)
V1	Encoding and Sanitization	5	1.9%
V2	Validation and Business Logic	4	1.5%
V3	Web Frontend Security	7	2.7%
V4	API and Web Service	4	1.5%
V5	File Handling	4	1.5%
V6	Authentication	8	3.0%
V7	Session Management	6	2.3%
V8	Authorization	4	1.5%
V9	Self-contained Tokens	2	0.8%
V10	OAuth and OIDC	31	11.8%
V11	Cryptography	18	6.8%
V12	Secure Communication	9	3.4%
V13	Configuration	16	6.1%
V14	Data Protection	10	3.8%
V15	Secure Coding and Architecture	14	5.3%
V16	Security Logging and Error Handling	21	8.0%
V17	WebRTC	10	3.8%
Total	—	**173**	**65.8%**

The unmapped ASVS Level 2 controls presented in Table 2 are not evenly distributed across domains. Instead, the largest concentration of gaps is found in the following domains:

- V10 – OAuth and OIDC (31 controls)
- V16 – Security Logging and Error Handling (21 controls)
- V11 – Cryptography (18 controls)
- V13 – Configuration (16 controls)
- V15 – Secure Coding and Architecture (14 controls)

These domains include requirements related to:

- Token validation, PKCE, audience restriction and delegated authorization decisions (V10)
- Log integrity, structured security event logging, secure log storage and failure handling (V16)
- Cryptographic key lifecycle management, crypto-agility, entropy requirements and algorithm strength (V11)
- Secret management, least privilege between backend components and secure production configuration (V13)
- Secure dependency management, mass assignment protection, type safety and resilience against resource exhaustion (V15).

This distribution indicates that while MFN defines several architectural and governance-level security requirements, it does not systematically implement control-level, protocol-specific or runtime security verification requirements defined in ASVS.

2.2.5.5 High-Risk Gaps

Although not all unmapped controls presented in Table 2, represent equal security risk, the most significant gaps are concentrated in the following domains:

OAuth and OIDC Security (V10)

ASVS 5.0 defines detailed and testable requirements for OAuth and OIDC implementations, including PKCE enforcement, token audience validation, ID token

replay protection, scope restriction, refresh token revocation and authorization server validation. MFN does not define protocol-specific requirements for secure OAuth/OIDC implementations. As a result, applications could formally comply with MFN while still misconfiguring token validation or authorization flows.

Security Logging and Error Handling (V16)

ASVS requires structured logging with defined metadata, protection against log tampering, secure log transmission, logging of failed authorization attempts and safe error handling. MFN mandates logging in principle but does not specify detailed security-event coverage, log integrity guarantees, or strict failure-handling requirements. This creates a gap in detection and forensic readiness.

Cryptography (V11)

ASVS defines strict requirements for:

- Key lifecycle management
- Cryptographic inventory
- Crypto agility
- Minimum security strength
- Approved algorithms
- CSPRNG entropy requirements

MFN refers to cryptographic compliance at a general level but does not define detailed key lifecycle controls or crypto-agility requirements.

Configuration and Secret Management (V13)

ASVS specifies detailed requirements for:

- Secret management vault usage
- Least privilege between services
- Prohibition of default credentials
- Allowlisting outbound connections
- Hardening production configurations

MFN addresses configuration principles but does not consistently define verifiable control-level requirements for backend service authentication and secret lifecycle management.

2.2.5.6 Interpretation of Results

The findings confirm that MFN is not designed to function as a comprehensive application security verification standard equivalent to ASVS Level 2. MFN operates as an organizational minimum requirement framework, primarily addressing architectural, cryptographic, deployment and governance-related aspects.

In contrast, ASVS Level 2 functions as a detailed application security verification framework with explicitly defined, testable and protocol-specific controls.

The absence of specific ASVS controls in MFN does not automatically imply insecure systems. Rather, it indicates that MFN alone does not provide systematic, verifiable assurance of application-level security controls.

2.2.5.7 Summary

The comparison between MFN and OWASP ASVS 5.0 Level 2 demonstrates that while MFN covers essential baseline security requirements, it leaves a substantial number of high-risk application security controls unaddressed.

MFN cannot be considered equivalent to ASVS Level 2 as a comprehensive application security standard. **To achieve structured and measurable application-level security assurance, MFN should be complemented with ASVS or a similar detailed application security framework**, particularly in the domains of:

- OAuth and OIDC security
- Cryptographic key lifecycle management
- Secure configuration and secret management
- Secure coding practices
- Security logging and failure handling

These findings provide a structured basis for the subsequent empirical system-level audit presented in Section 2.2.7.

2.2.6 RFN audit

2.2.6.1 Objective and Scope

This section evaluates the alignment between the RIA Cross-Functional Requirements (CFR) and OWASP Application Security Verification Standard (ASVS) 5.0 Level 2. The purpose of the audit is to determine to what extent CFR provides verifiable coverage of ASVS Level 2 controls within the context of public-sector software governance. The assessment covers all 263 ASVS 5.0 Level 2 controls (V1–V17).

2.2.6.2 Methodology

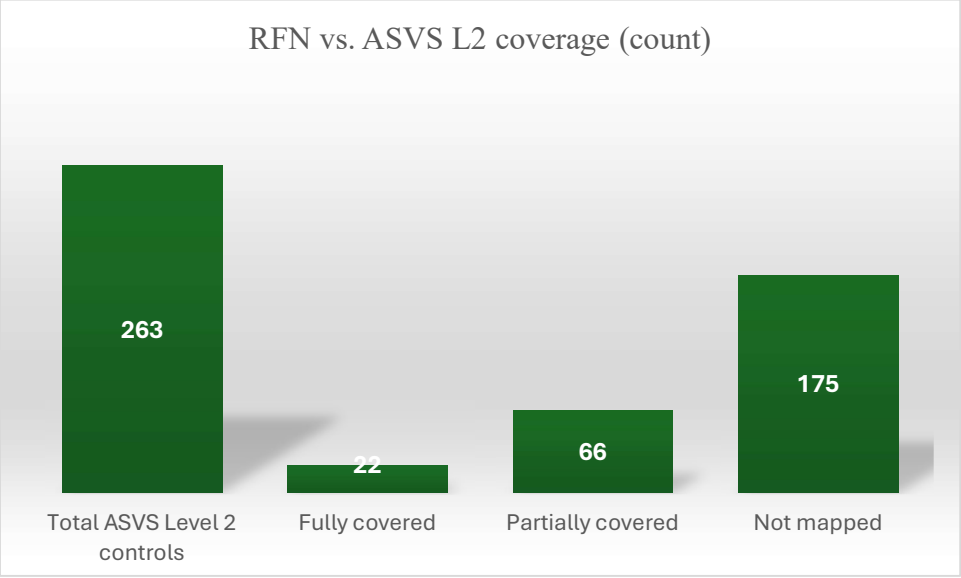
The mapping procedure is described in paragraph 2.1.4.

2.2.6.3 Quantitative Results

The mapping results are summarized in Table 3:

Total ASVS Level 2 controls: 263 (Fully 22, Partial 66, Not mapped 175).

Table 3 RFN vs. ASVS L2 coverage.



This means that CFR provides 8% full control-level coverage, 25% partial conceptual coverage and 67% without direct mapping.

While approximately one-third of ASVS Level 2 controls show some degree of alignment (Full + Partial ≈ 33%), only a small fraction meets the criteria for direct and verifiable

equivalence. These figures presented in Table 3, indicate that CFR does not function as a control-level application security standard.

2.2.6.4 Nature of coverage

The control-level analysis shows that the limited number of fully covered ASVS Level 2 controls within CFR is concentrated in foundational security domains. These include:

- mandatory use of TLS for secure communication
- prohibition of storing secrets in source code repositories
- requirement for static code analysis
- security testing prior to production deployment
- adherence to national cryptographic guidance
- baseline logging requirements

These controls represent governance-enforceable technical safeguards that CFR explicitly mandates. However, like the MFN analysis, the largest structural gaps in CFR appear in the same ASVS domains that require application-internal behavioural security controls. Significant portions of the following chapters remain unmapped or only partially covered:

- V10 – OAuth and OIDC (31 unmapped controls)
- V11 – Cryptography (18 unmapped controls)
- V13 – Configuration (16 unmapped controls)
- V16 – Security Logging and Error Handling (21 unmapped controls)

In these domains, CFR either remains silent or formulates high-level architectural expectations without defining verifiable control criteria. For example:

- CFR promotes centralized identity management but does not define token validation rules, scope enforcement, or audience restrictions required by ASVS.
- CFR encourages REST-based APIs but does not prescribe object-level authorization or excessive data exposure prevention mechanisms.
- CFR requires logging but does not define integrity protection, event correlation, or active monitoring thresholds.

- CFR defines general architectural robustness but does not regulate business logic abuse prevention (e.g., race conditions, workflow bypass, privilege escalation through logic flaws).

Thus, while thematic alignment exists, most mapped controls fall into the partial category. **CFR articulates expected secure outcomes but does not:**

- **define measurable verification criteria**
- **specify structured abuse-case testing requirements**
- **prescribe fine-grained control enforcement mechanisms**
- **require CI/CD-verifiable evidence for control implementation.**

This structural characteristic distinguishes CFR from ASVS Level 2. ASVS defines detailed, granular and testable control statements focused on application behaviour, whereas CFR operates at a governance and architectural principal level.

2.2.6.5 Interpretation

The audit confirms that CFR is not designed to operate as an application security verification framework equivalent to ASVS Level 2.

CFR primarily establishes governance principles, architectural direction, lifecycle expectations, compliance-level safeguards.

In contrast, ASVS Level 2 defines: detailed, behaviour-oriented, technically verifiable application security controls.

Therefore, **the relatively low percentage of full control coverage does not imply inadequacy of CFR within its intended purpose.** Rather, it reflects its higher abstraction level. CFR operates at the **policy and architecture layer**, whereas ASVS operates at the **application verification layer.**

2.2.6.6 Summary

Approximately two-thirds of ASVS Level 2 controls lack direct representation in CFR.

CFR cannot be considered equivalent to ASVS Level 2 as a verifiable application security standard. Instead, it functions as a governance and architectural baseline that must be complemented by more granular and testable security control frameworks.

2.2.7 Public faced system audit

2.2.7.1 Objective and Scope

This section presents the empirical application-level security assessment of one public-facing information system developed under the institutional requirement framework described earlier. The objective of this audit was to evaluate:

- The practical implementation of OWASP ASVS Level 2 controls
- The availability of verifiable technical evidence
- The effectiveness of implemented security mechanisms

2.2.7.2 Assessment Method

The system-level security assessment was conducted using the procedure described in Section 2.1.5. The audit was performed by an independent security testing specialist using a structured white-box methodology aligned with OWASP ASVS Level 2 control domains. The evaluation was based on a documented security assessment report and included control-level verification of implemented security mechanisms.

Testing was performed under the following conditions:

- white-box access to application source code,
- HTTP-level access to the deployed application,
- no SSH access to the underlying server,
- no direct access to the production database,
- no direct access to server-side log files.

The assessment included manual expert-driven testing, static code inspection, dynamic analysis of application behavior and validation of abuse cases such as injection, authorization bypass and improper input handling.

Findings were classified according to ASVS control reference, technical risk level, and remediation status. The purpose of the audit was not to evaluate the overall quality of a single system, but to determine whether requirement-level gaps identified in the document analysis are also observable at implementation level.

2.2.7.3 High-Level Results

The main findings were:

- Multiple high-risk findings
- Additional findings introduced after architectural modifications
- Limited effective remediation
- Regression issues after security control additions

The overall production readiness assessment was negative.

Key observations:

- Remote code execution became achievable due to:
  - Use of an outdated framework version,
  - Unsafe deserialization (unsterilized () on user-controlled input).
- Sensitive credentials (API keys, FTP access data) were found in source code.
- Stored HTML injection vulnerabilities were identified in multiple views.
- File handling controls were insufficient.
- Server configuration weaknesses were present.
- HTTP method restrictions were not enforced server-side.
- Internal network information was disclosed via error messages.
- Multiple endpoints returned HTTP 500 errors.
- Application functionality was degraded after introducing a Web Application Firewall (WAF).

The findings demonstrate that the identified weaknesses were not isolated defects but reflected systemic deficiencies in secure coding practices, dependency management, and runtime control enforcement.

2.2.7.4 Risk Distribution

Findings were distributed across multiple ASVS 5.0 domains:

- V1 – Encoding and Sanitization
- V2 – Validation and Business Logic
- V4 – API and Web Service
- V5 – File Handling

- V6 – Authentication
- V7 – Session Management
- V8 – Authorization
- V11 – Cryptography
- V12 – Secure Communication
- V13 – Configuration
- V14 – Data Protection
- V15 – Secure Coding and Architecture
- V16 – Security Logging and Error Handling

2.2.7.5 Summary of Security Findings by Risk Level:

Table 4 Summary of Security Findings by Risk Level.

Risk Level	**Primary Vulnerability Types Identified**	**Structural Security Implication**
Critical	Unsafe deserialization enabling remote code execution; unsupported framework version	Full compromise potential at application level; systemic architectural risk
High	Stored injection vulnerabilities; exposed credentials in source code; insecure direct access to resources	Privilege escalation, data manipulation, confidentiality breach
Medium	Insufficient HTTP method restrictions; incomplete server-side validation; configuration weaknesses	Increased attack surface; bypass of intended control logic
Low	Incomplete security headers; functional instability (HTTP 500 responses); minor logging weaknesses	Defense-in-depth degradation; reconnaissance facilitation
Informational	Internal network information disclosure; analytics data exposure; usability-linked misconfigurations	Information leakage; indirect risk amplification

2.2.7.6 Structural Weakness Patterns

The findings presented in Table 4 indicate recurring structural weaknesses:

1. **Legacy Framework Dependency.** The application was built on an unsupported framework version, introducing systemic risk beyond individual control failures.
2. **Insecure Deserialization.** User-controlled data reaching unserialize() created potential full system compromise scenarios.
3. **Injection Exposure.** Multiple stored HTML injection vectors indicate insufficient output encoding enforcement.
4. **Configuration and Infrastructure Gaps.** Improper HTTP method restriction, internal IP leakage, and missing strict CSP enforcement demonstrate configuration-level control weaknesses.
5. **Incomplete Security Control Enforcement.** While certain baseline measures were present (e.g., TLS, some headers), enforcement was inconsistent and lacked depth.
6. **Ineffective Compensating Controls**. The addition of a Web Application Firewall mitigated only one previously identified issue and introduced functional instability. This demonstrates that compensating perimeter controls cannot replace secure application design.

2.2.7.7 Production Readiness Evaluation

The independent assessment concluded that the system was not production ready.

The primary reasons included:

- Achievable remote code execution
- Unsupported framework version
- Multiple unresolved injection vulnerabilities
- Functional instability (HTTP 500 responses)
- Weak abuse-case protection
- Limited effective remediation after initial findings.

From a governance perspective, this demonstrates that compliance with institutional requirement frameworks does not automatically translate into ASVS Level 2 conformity or production-grade security assurance.

2.2.7.8 Interpretation

This empirical case demonstrates three important insights:

- High-level requirement frameworks (CFR) are insufficient for application-level security verification.
- Even detailed institutional requirements (MFN) do not guarantee behavioural security correctness.
- ASVS-based verification exposes control-level weaknesses that governance documents alone cannot detect.

The findings reinforce the central thesis argument:

Systematic mapping and verification against a detailed application security standard (such as OWASP ASVS Level 2) is necessary to ensure measurable and enforceable security assurance in public-sector software development.

3 Results

The mapping results obtained using the deductive standards-based document analysis described in Chapter 2 show that neither CFR nor MFN provides coverage comparable to OWASP ASVS 5.0 Level 2 when evaluated as a control-level verification standard. However, the nature of alignment differs substantially due to the different abstraction levels and normative purposes of the two documents.

MFN provides partial or full alignment with **90 of 263** ASVS Level 2 controls **(≈ 34%**), while **173 controls (≈ 66%)** remain without a direct equivalent. This indicates that MFN captures a limited subset of application security requirements, primarily those that can be expressed as institutional implementation rules and verified through development and deployment practices.

CFR demonstrates a similar overall proportion of some alignment, but with a markedly different structure. Out of 263 ASVS Level 2 controls, the control-level mapping classified **22 controls as fully covered**, **66 as partially covered** and **175 as not mapped**. In other words, CFR provides approximately 8% direct and verifiable control-level coverage, 25% conceptual or outcome-level alignment and 67% without any explicit mapping. The low share of full mappings confirms that CFR is not formulated as a verifiable application security control catalogue, but rather as a governance and architecture baseline.

Taken together, these results demonstrate that the similarity in overall mapped share (roughly one-third) does not imply functional equivalence. MFN contributes comparatively more enforceable technical constraints, whereas CFR aligns mainly at the level of principles and expected outcomes.

Across both mappings, missing or only partially addressed controls were not evenly distributed across ASVS chapters. Instead, gaps were concentrated in domains that require behavioral correctness, abuse-case resilience, and application-level verification beyond baseline governance requirements.

In the MFN mapping, the highest concentrations of unmapped controls were found in:

- V10 – OAuth and OIDC (31 unmapped controls)

- V11 – Crypthography (18 unmapped controls)
- V13 – Configuration (16 unmapped controls)
- V16 – Security Logging and Error Handling (21 unmapped controls)

The CFR mapping exhibited the same general pattern. Where ASVS defines detailed requirements for token handling, workflow integrity, object-level authorization, security monitoring and detection-oriented logging, CFR either provides only high-level expectations or does not cover at all.

This clustering of gaps is analytically important because the affected domains correspond to widely exploited attack paths in real-world systems. The results indicate that both documents tend to regulate baseline security hygiene more strongly than behaviour-driven weaknesses such as business logic abuse, fine-grained authorization in APIs and operational detection capabilities.

The analysis shows that the fully covered controls in CFR are mainly concentrated in foundational governance-enforceable areas, such as:

- mandatory secure communication (TLS)
- prohibition of hard-coded secrets
- static code analysis expectations
- security testing before production release
- cryptographic compliance with national guidance
- baseline logging requirements.

These represent controls that can be mandated at procurement or governance level and verified through process artefacts.

However, the majority of mapped ASVS controls in CFR fall into the Partial category. In these cases, CFR expresses expected outcomes but does not define the control in a way that would satisfy ASVS verification criteria. CFR typically does not:

- define measurable verification criteria
- prescribe abuse-case testing requirements
- specify control-level enforcement mechanisms
- require CI/CD-verifiable evidence for implementation.

This distinguishes CFR structurally from ASVS Level 2. ASVS control statements are designed to be directly testable and to validate application behaviour under realistic attack conditions, whereas CFR remains primarily outcome-oriented and architecture-governance focused.

MFN shows a stronger tendency toward implementable and verifiable controls than CFR, but the mapped controls still represent a limited subset of ASVS Level 2. MFN more often specifies “how” a requirement should be met, but it still does not systematically cover the ASVS domains that require explicit abuse-case verification and behavioural security constraints.

The applied descriptive ASVS-based assessment of one public-facing system demonstrated security weaknesses across multiple ASVS domains and concluded that the system was not production ready. The findings included high-severity weaknesses with potential for systemic compromise, as well as multiple supporting weaknesses related to configuration robustness, access control enforcement and error handling.

The results also showed that compensating controls introduced after the initial assessment provided limited mitigation value and caused functional regressions. This indicates that reactive add-on controls cannot substitute for secure design and secure-by-default implementation practices when core application-level weaknesses remain unaddressed.

Overall, the empirical results provide a practical illustration that compliance with governance and institutional requirement frameworks does not guarantee ASVS Level 2 conformity at application level.

The results of this thesis show that CFR and MFN function as complementary governance and institutional control frameworks, but neither provides comprehensive or verifiable coverage equivalent to OWASP ASVS Level 2. CFR aligns primarily at the level of principles and expected outcomes, resulting in low full control-level coverage. MFN provides a more enforceable control set, but still leaves major parts of ASVS Level 2 unregulated.

The empirical ASVS-based assessment supports these findings by demonstrating that application-level weaknesses can persist despite adherence to governance and institutional requirements. This establishes the basis for the discussion in Chapter 5

regarding the need for an explicit verification-layer standard within public-sector secure development governance.

4 Discussion

The results presented in Chapter 3 demonstrate that the core issue identified in this thesis is not the existence of insecure requirements, but rather the **absence of a clearly defined and formally integrated verification layer within the public-sector secure development governance model.**

The results presented in Chapter 3, obtained through deductive document analysis and system-level security assessment, demonstrate that the core issue identified in this thesis is not the existence of insecure requirements. The main problem is the **absence of a clearly defined and formally integrated verification layer within the public-sector secure development governance model.**

CFR and MFN were not designed to function as comprehensive application security verification standards. Instead, they operate at different abstraction levels and fulfil different normative purposes. However, when evaluated against OWASP ASVS Level 2, the analysis reveals that the current layered relationship between governance principles and application-level behavior is incomplete.

The mapping results show that CFR and MFN align with approximately one-third of ASVS Level 2 controls. However, the structural nature of this alignment differs fundamentally from ASVS.

CFR primarily defines:

- strategic security expectations
- architectural direction
- compliance-level safeguards
- high-level outcome-oriented obligations.

MFN translates selected CFR expectations into:

- institutional implementation constraints
- technical and CI/CD-enforceable controls
- operational procedures.

ASVS, in contrast, defines granular, behavior-oriented, testable, abuse-case-driven application security controls.

The empirical findings demonstrate that governance compliance does not automatically imply behavioral correctness at application level. While CFR and MFN may mandate secure communication, code review or logging presence, they do not systematically require verification of business logic abuse resistance, token misuse prevention, object-level authorization enforcement or security event correlation capabilities.

The central conclusion is therefore not that CFR or MFN are inadequate within their intended scope, **but the current governance structure lacks an explicit and mandatory verification-layer mechanism comparable to ASVS Level 2.**

Without such a layer, application-level security assurance becomes dependent on:

- contractor competence
- ad hoc security testing
- interpretation of high-level requirements
- variable maturity of development partners.

This structural gap explains why empirical weaknesses emerged in domains that were only partially covered or conceptually addressed at requirement level.

The findings suggest that secure development governance in the public sector should explicitly distinguish between three layers:

- Policy and Architecture Layer- defining strategic and regulatory expectations (CFR).
- Implementation and Enforcement Layer- defining institutional, operational and CI/CD-enforceable requirements (MFN).
- Verification and Assurance Layer- defining control-level behavioural validation criteria (e.g., ASVS Level 2).

At present, the first two layers are formally defined, while the third layer is implicitly assumed, but not systematically embedded into procurement, development acceptance, and operational governance processes.

The study supports the proposition that ASVS Level 2 or a tailored equivalent control catalogue should be formally integrated into:

- procurement documentation
- acceptance criteria for public-facing systems
- structured security verification processes
- CI/CD-based evidence collection mechanisms.

In practice, such integration would not necessarily require replacing existing governance frameworks. **A realistic implementation approach would be to** reference ASVS Level 2 selectively within procurement technical specifications, supplier acceptance criteria and security testing requirements for public-facing web applications and APIs. Under this model, CFR would continue defining governance expectations and MFN implementation constraints, while ASVS would function as the formal verification baseline for application-level security controls. Verification evidence could be produced through a combination of manual security assessment, CI/CD-integrated testing and structured compliance reporting.

4.1 Methodological Reflection and Limitations

This study has several limitations that should be acknowledged.

First, the research is based on an embedded single-case study within one public-sector organization. The results therefore provide contextual and analytical insight but cannot be generalized statistically to all institutions. The purpose of the study is analytical generalization, not statistical generalization, meaning that the findings are intended to illustrate structural characteristics of the requirement model rather than to represent all public-sector systems.

Second, the requirement mapping process (Full / Partial / Not mapped) is based on deductive control-level classification. Although the mapping followed predefined criteria and was applied consistently, the interpretation of requirement equivalence involves expert judgement. This limitation was mitigated by using a fixed analytical benchmark (ASVS Level 2), domain-level aggregation and comparison across multiple data sources.

Third, the empirical part of the study relies on an independent white-box security assessment report. The testing conditions allowed access to application source code and HTTP interface, but did not include direct access to the server, database, or log files. As a result, some ASVS controls could not be fully verified at runtime, which may have affected the completeness of the system-level findings.

Fourth, the security assessment was originally conducted using an earlier ASVS-based testing structure, while the present study applies ASVS 5.0 Level 2 as the analytical benchmark. Although control domains were mapped to the newer version, minor differences between versions may influence domain-level classification.

Despite these limitations, the combination of deductive document analysis, domain-level gap clustering and empirical system-level assessment provides methodological triangulation, which strengthens construct validity and supports the robustness of the conclusions.

However, the structural characteristics identified in this study are unlikely to be unique to the selected organization. Many Estonian public-sector institutions rely on the same CFR baseline and use institution-specific requirement frameworks derived from similar governance principles. Because the identified gaps were concentrated in application-level behavioural control domains rather than organization-specific implementation details, the findings may reflect broader structural limitations within public-sector secure development governance models.

4.2 Recommendations and Future Development Path

Based on the findings of this study, several improvements can be proposed to strengthen secure software governance in the public sector.

In the short term, institutional development processes should introduce explicit application-level verification requirements for public-facing systems. In particular, the use of a standardized control checklist aligned with ASVS Level 2 would allow security controls to be verified in a consistent and measurable manner. In addition, procurement, development and acceptance procedures should require verifiable technical evidence,

such as CI/CD security outputs, authorization test results, and logging validation records, especially for high-risk security domains.

In the medium term, security control validation should be integrated directly into development pipelines. This includes automated verification of configuration settings, structured abuse-case testing for business logic and API-level authorization, and defined requirements for monitoring, log integrity, and event correlation. Such measures would reduce reliance on manual interpretation of requirements and improve repeatability of security assurance.

In the longer term, the results suggest the need for a more formal connection between national-level development requirements and application-level security verification standards. One possible approach would be the creation of a public-sector ASVS profile aligned with existing CFR and institutional requirement frameworks. In addition, a separate verification-layer governance model could be defined to bridge high-level policy requirements and control-level security validation. Moving toward compliance-as-code and automated verification mechanisms would further support consistent, auditable, and repeatable security assurance.

However, the findings also indicate that automation alone is insufficient for comprehensive application-level security verification. Several identified weaknesses, particularly those related to business logic abuse, insecure workflows and authorization behavior, required contextual manual analysis and abuse-case testing. Therefore, automated compliance mechanisms should be combined with expert-driven security assessment rather than treated as a complete replacement for manual verification.

Overall, these steps would shift the current governance model from principle-based guidance toward a measurable and verifiable secure development framework.

4.3 Conclusion

The discussion confirms that the core issue identified in this thesis is structural rather than documentary. The analysis shows that CFR and MFN are not deficient within their intended governance scope. Instead, the limitation lies in the absence of a formally

defined verification-layer framework that would translate governance requirements into control-level, testable security criteria.

The results obtained through deductive document analysis and system-level security assessment indicate that compliance with governance and institutional requirement frameworks does not guarantee application-level security assurance. The observed correspondence between unmapped ASVS controls and empirical vulnerabilities suggests that missing behavioral and control-level requirements represent a systematic risk rather than isolated implementation errors.

The findings support the interpretation that public-sector secure software governance currently operates effectively at policy and implementation levels, but lacks a consistent mechanism for verifying application behavior against a recognized security control standard.

Therefore, secure software development in the public sector should incorporate an explicit verification-layer framework, such as OWASP ASVS Level 2 or an equivalent control catalogue, to ensure that security requirements are not only defined and implemented, but also demonstrably validated through repeatable and measurable verification procedures.

This conclusion forms the basis for the final chapter, which summarizes the main contributions of the thesis and provides a direct answer to the research question.

Summary

In recent years, secure software development has become increasingly critical in the public sector [32] due to the growing frequency and impact of cyber incidents. Public-facing information systems process personal data, support administrative decision-making and integrate with national digital infrastructure. Ensuring that such systems are developed with sufficient and verifiable security controls is therefore essential for both operational reliability and public trust.

The objective of this thesis was to determine whether the Estonian public-sector Cross-Functional Requirements (CFR) and institutional Non-Functional Requirements (MFN) provide sufficient and verifiable coverage when evaluated against the OWASP Application Security Verification Standard (ASVS) Level 2. The study used an embedded qualitative case study design combining deductive standards-based requirement mapping with an empirical ASVS-based security assessment of one public-facing system.

The requirement-level analysis showed that MFN provides alignment (Full or Partial) with 90 of 263 ASVS Level 2 controls (approximately 34%), while 173 controls remain unmapped. CFR demonstrated 22 fully covered controls, 66 partially covered controls, and 175 controls without direct mapping. Although both frameworks showed roughly one-third conceptual alignment with ASVS Level 2, the nature of this alignment differs significantly. CFR operates primarily at the governance and architectural level, expressing outcome-oriented expectations, whereas MFN introduces more implementation-oriented and enforceable technical constraints. Neither framework, however, provides comprehensive behavioral control coverage comparable to ASVS Level 2.

The analysis further demonstrated that coverage gaps are concentrated in specific domains, particularly OAuth and token handling, business logic security, API-level authorization, configuration robustness and logging and monitoring controls. These domains require granular, testable and abuse-case-oriented verification criteria that are not systematically defined in CFR or MFN.

The empirical ASVS-based assessment of a public-facing system supported these findings. The system exhibited weaknesses across multiple ASVS domains and was assessed as not production ready. Importantly, the observed weaknesses corresponded to the same control domains identified as weakly covered in the requirement mapping. This triangulation strengthens the conclusion that requirement-level incompleteness is not merely theoretical but may translate into practical application-level risk.

The central conclusion of this thesis is that CFR and MFN function as complementary governance and institutional enforcement frameworks, but they do not constitute a comprehensive application security verification standard. Compliance with governance requirements does not automatically imply conformity with ASVS Level 2. The absence of an explicit verification layer within the secure development governance model represents a structural gap.

The study therefore proposes a layered secure development model consisting of:

- a policy and architectural layer (CFR),
- an implementation and enforcement layer (MFN),
- a verification and assurance layer (e.g., ASVS Level 2 or a tailored equivalent).

Integrating an explicit verification layer into procurement, development acceptance, and CI/CD evidence collection processes would enable measurable, repeatable, and auditable application-level security assurance.

This thesis contributes:

- a structured control-level mapping of CFR and MFN against ASVS Level 2,
- a domain-level gap profile highlighting risk-concentrated areas,
- empirical validation that requirement gaps correspond to observed application weaknesses,
- a layered governance interpretation for public-sector secure development.

Future research could extend the analysis across multiple institutions and systems, introduce inter-rater validation of requirement mappings and explore compliance-as-code mechanisms capable of producing automated ASVS-aligned assurance evidence.

Ensuring secure software development in the public sector requires not only defining what must be secure and how development should be conducted, but also systematically verifying how applications behave under realistic threat conditions. The findings further suggest that future public-sector secure development models should combine governance frameworks, automated compliance mechanisms and structured manual verification practices rather than relying exclusively on any single assurance approach. Introducing a structured verification layer aligned with ASVS Level 2 would significantly improve the integrity, resilience, and trustworthiness of public digital services.

References

[1] ENISA. (2025). *ENISA Threat Landscape 2025.*

[2] “Microsoft Security Development Lifecycle.” https://www.microsoft.com/en-us/securityengineering/sdl/ (accessed Feb. 02, 2026).

[3] M. Stoica, M. Mircea, and B. Ghilic-Micu, “Software development: Agile vs. traditional.,” Informatica Economică, vol. 17, no. 4, 2013, doi: 10.12948/issn14531305/17.4.2013.06.

[4] Diao, O. (2025). *Security-by-5D: A new holistic secure model. SSRN.* https://doi.org/10.2139/ssrn.5392978

[5] Fucci, D. (2024). *Evaluating software security maturity using OWASP SAMM. Information and Software Technology.*

[6] Wuyts, K., Scandariato, R., & Joosen, W. (2023). *Empirical evaluation of maturity models for software security assurance. Journal of Systems and Software, 198*, 111562.

[7] Open Web Application Security Project, “Application Security Verification Standard OWASP Application Security Verification Standard (ASVS) | OWASP Foundation (accessed Feb. 22, 2026).

[8] “IBM. What is DevSecOps?.” https://www.ibm.com/think/topics/devops-lifecycle#498277090 (accessed Feb. 02, 2026).

[9] Lee, K., & Park, S. (2025). *Policy-as-code and compliance automation in DevSecOps pipeline. Journal of Software Engineering and Applications.*

[10] Smith, A., & Jones, B. (2022). *Compliance-as-code: A framework for regulatory automation in CI/CD. Journal of Engineering, Science and Cybernetics.*

[11] “ISO - ISO/IEC 27001 — Information security management.” https://www.iso.org/isoiec-27001-information-security.html (accessed Feb. 22, 2026).

[12] Shin, Y., & Williams, L. (2020). *Toward more effective secure software development with DevSecOps. IEEE Software, 37*(2), 55–61.

[13] Rahman, M. M., & Williams, L. (2021). *Characterizing secure DevOps practices in industry: An empirical study. Information and Software Technology, 135*, 106552.

[14] U.S. Department of Defense, Chief Information Officer. (2024). *State of DevSecOps.*

[15] “Software Bill of Materials (SBOM) definition “https://csrc.nist.gov/glossary/term/software_bill_of_materials (accessed Feb. 22, 2026).

[16] Xia, B., et al. (2023). *An empirical study on SBOMs. Proceedings of the International Conference on Software Engineering (ICSE 2023).*

[17] Nocera, S., et al. (2025). *Adoption of SBOMs in open-source software ecosystems. Information and Software Technology.*

[18] Papaioannou, T., et al. (2025). *Software bill of materials in software supply chain security: A systematic literature review. arXiv preprint arXiv:2506.03507.*

[19] Zhang, Y., et al. (2025). *Ensuring authentication and authorization security using OWASP ASVS controls in enterprise applications. Journal of Information Security and Applications.*

[20] Yin, R. K. (2018). *Case study research and applications: Design and methods* (6th ed.). Sage Publications.

[21] Creswell, J. W., & Poth, C. N. (2018). *Qualitative inquiry and research design: Choosing among five approaches* (4th ed.). Sage Publications.

[22] RIA. (n.d.). *Cross-Functional Requirements (CFR)*. Estonian Government Code Repository. https://koodivaramu.eesti.ee/e-gov/cfr (accessed Feb 2026).

[23] Transport Administration. (n.d.). *MFN – Non-Functional Requirements documentation*. MFN.md · main · TRAM-public / MFN · GitLab (accessed Feb, 2026).

[24] OWASP Foundation. (n.d.). OWASP risk rating methodology. https://owasp.org/www-community/OWASP_Risk_Rating_Methodology (accessed Feb, 2026).

[25] RIA. (n.d.). *Tasks and structure of the authority*. Estonian Information System Authority. https://www.ria.ee/en/authority-news-and-contact/authority-and-management/tasks-and-structure-authority (accessed Feb 2026).

[26] Estonian Information System Authority. (n.d.). *RIHA – The Estonian national information system registry*. https://www.riha.ee/Avaleht (accessed Feb, 2026).

[27] Estonian Transport Administration. (n.d.). *About the Estonian Transport Administration*. https://www.transpordiamet.ee/en/administration-news-and-contact/administration (accessed Feb 2026).

[28] Estonian Transport Administration. (n.d.). *Traffic Register (Liiklusregister)*. https://www.transpordiamet.ee/liiklusregister

[29] Estonian Transport Administration. (n.d.). *Electronic Maritime Information System 2.0*. https://www.transpordiamet.ee/elektrooniline-mereinfosusteem-20

[30] Zhang, L., et al. (2025). *A landscape study of open-source tools for software bill of materials (SBOM). ICSE Workshop on Software Supply Chain Security.*

[31] Umeugo, W. (2023). Secure software development lifecycle: A case for adoption in software SMEs. International Journal of Advanced Research in Computer Science. Retrieved from https://www.researchgate.net/publication/368737283_SECURE_SOFTWARE_DEVELOPMENT_LIFECYCLE_A_CASE_FOR_ADOPTION_IN_SOFTWARE_SMES

[32] RIA (2026). RIA Cybersecurity Yearbook 2026 (RIA kuberturvalisuse aastaraamat 2026). https://www.ria.ee/sites/default/files/documents/2026-02/RIA-kuberturvalisuse-aastaraamat-2026.pdf

[32] OWASP Foundation. (n.d.). OWASP Application Security Verification Standard (ASVS) 5.0 (Version 5.0) [Source code]. GitHub. https://github.com/OWASP/ASVS/tree/master/5.0 (accessed Jan 2026).

[33] OWASP Foundation. (2025, May). *OWASP Application Security Verification Standard (ASVS) version 5.0.0*. OWASP Foundation (accessed Jan 2026).

[34] Alhazmi, O. H., & Malaiya, Y. K. (2022). *Empirical analysis of software vulnerability discovery models in secure development lifecycles. Empirical Software Engineering, 27*(4), 87.

[35] Chen, M., et al. (2025). *Policy-driven SBOM on GitHub: An empirical study. arXiv preprint arXiv:2509.01255.*

[36] Dagstuhl Reports (2023). *Empirical evaluation of secure development processes.*

[37] De Win, B., Scandariato, R., Preneel, B., & Joosen, W. (2009). *On the secure software development process: CLASP, SDL, etc. Information and Software Technology, 51*(6), 1192–1207.

[38] Steinhardt, I. *Deductive qualitative content analysis*. In I. Steinhardt (Ed.), Qualitative content analysis. Routledge, 2026

[39] Lam, H., & Smith, J. (2023). *Empirical evaluation of secure development processes. Dagstuhl Reports, 13*(5), 1–23.

[40] T. W. Edgar and D. O. Manz, *Research Methods for Cyber Security*. Rockland, MA, UNITED STATES: Syngress, 2017.

[41] Elo,S.,&Kyngäs,H.(2008). The qualitative content analysis process. Journal of Advanced Nursing, 62(1), 107–115. https://doi.org/10.1111/j.1365-2648.2007.04569.x

[42] Bowen, G. A. (2009). Document analysis as a qualitative research method. Qualitative Research Journal, 9(2), 27–40. https://doi.org/10.3316/QRJ0902027

[43] Mirtsch, M., Kinne, J., & Blind, K. (2020). Exploring the adoption of the international information security management system standard ISO/IEC 27001: A web mining-based analysis. IEEE Transactions on Engineering Management.

[44] RIA. (n.d.). E-ITS Framework. https://eits.ria.ee/et (accessed Feb. 22, 2026).

[45] NIST Framework. Cybersecurity Framework | NIST (accessed Feb. 22, 2026).

[46] NIST SP 800-53 Rev. 5. SP 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations | CSRC (accessed Feb. 22, 2026).

[47] OWASP Foundation. (2020). *OWASP Software Assurance Maturity Model (SAMM) Version 2.0 diagram* [Diagram]. OWASP Foundation. https://owaspsamm.org/img/pages/SAMM_v2_diagram.svg

Appendix 1 – Cross-Functional Requirements (CFR)

Requirement	Extensibility	Category
Application code is version-controlled using Git.	Mandatory	Development
Application source code is clearly written and maintainable.	Mandatory	Development
Application software must be licensed.	Mandatory	Development
Application builds must not depend on external availability.	Expected	Development
Programming languages must be in Top 25 TIOBE index.	Expected	Development
All components must have at least 5 years of remaining EOL.	Expected	Development
Recovery plans must exist for service applications.	Expected	Development
Reusable code must be stored in the national code repository.	Recommended	Development
Application must pass security testing before production.	Mandatory	Delivery
Application must not be deployed with known vulnerabilities.	Mandatory	Delivery
Code failing tests must not be deployed.	Mandatory	Delivery
Code must undergo four-eyes code review.	Expected	Delivery
Deployment must be automated.	Expected	Delivery
Applications must be automatically monitored in production.	Expected	Delivery
Blue-green deployment should be used.	Recommended	Delivery
Semantic versioning must be used.	Recommended	Delivery
Configuration and secrets must not be stored in source code.	Mandatory	Architecture
Application must not create a new identity system.	Mandatory	Architecture
Technical components must validate access rights.	Expected	Architecture
APIs must have automatically generated documentation.	Expected	Architecture
Applications follow domain-driven design and microservices principles.	Expected	Architecture
Inter-system data exchange must use X-Road.	Expected	Architecture
UI and service functionality must be separated and communicate via API.	Expected	Architecture
Technical components must expose REST APIs.	Expected	Architecture
Application must be cloud ready.	Expected	Architecture
Distributed components must not share the same database or filesystem.	Expected	Architecture
Database must not contain business logic.	Expected	Architecture
UI state must be client-side; services stateless.	Recommended	Architecture

Components must log correlation IDs.	Draft	Architecture
User functionality must be covered by automated tests.	Expected	Quality
Applications must pass load testing before production.	Expected	Quality
Complex internal functions must have unit tests.	Expected	Quality
Functional scope must be documented with user stories.	Recommended	Quality
Application activity must be logged and audit logs stored separately.	Mandatory	Security
URL must not contain personal data or session tokens.	Mandatory	Security
UI and components must communicate over TLS/SSL.	Mandatory	Security
Secrets must not be stored in code repositories.	Mandatory	Security
Application must be resilient to external failures.	Mandatory	Security
Static code analysis must be applied.	Expected	Security
Inputs must be validated and sanitized.	Expected	Security
Outputs must be sanitized before use.	Expected	Security
Use up-to-date cryptographic algorithms.	Expected	Security
Internal components communicate over TLS/SSL.	Expected	Security
WAF rules must be created during development.	Draft	Security
All activities must be traceable to individuals.	Draft	Security
MFA must apply to administrative functions.	Draft	Security
Data must be stored in UTF-8 or higher encoding.	Mandatory	Data Management
National datasets must be described in RIHA.	Mandatory	Data Management
Data must be exportable in machine- and human-readable formats.	Mandatory	Data Management
Logical deletion must be used instead of physical deletion.	Expected	Data Management
Time must follow ISO 8601 format.	Expected	Data Management
Objects must be identified using registry codes.	Expected	Data Management
Address data system requirements must be followed.	Expected	Data Management
Classifier system requirements must be followed.	Expected	Data Management
Use RIA data tracker.	Expected	Data Management
System must have a data description.	Expected	Data Management
Data quality model must be applicable.	Expected	Data Management

Data quality measurement results must be exportable.	Expected	Data Management
System must integrate and use official classifiers.	Expected	Data Management

Appendix 2 – Institutional Non-Functional Requirements (MFN)

Category	Requirement
meta	When applying requirements, consider the specific characteristics of the software.
meta	Apply requirements according to the hierarchy principle.
formatting	Use UTF-8 encoding in databases and applications.
formatting	Use consistent line-ending encoding within a single file.
formatting	Use RFC 3339 format for representing time values.
license	Software must be marked with an appropriate license.
module_structure	External dependencies must be clearly documented.
module_structure	The application must be resilient to failures of external systems.
module_structure	Public service interfaces must be clearly separated from non-public, internal, and configuration interfaces.
module_structure	HTTP REST APIs must be described in machine-readable OpenAPI format.
language	Source code and logs must be in English.
language	Estonian-language texts must comply with EVS 8:2008.
testing	Source code must include unit tests.
testing	Software must undergo security testing before production deployment.
testing	Automated tests must report results in both human- and machine-readable formats (e.g., JUnit XML and HTML).

testing	Automated tests must be executed using GitLab Runner.
code_quality	Code must pass static code analysis.
frontend	Style information must be placed in CSS files.
frontend	Use Sass for large stylesheets.
frontend	Follow HTML5 and CSS3 standards.
frontend	The application must function in browsers supporting the two latest versions of eID base software.
frontend	If browser support is missing, the application must display an error message.
usability	The web application user interface must meet WCAG 2.2 Level AA accessibility requirements.
URLs	Use clear, consistent, human-readable URLs.
URLs	Each page must have a unique URL.
URLs	URLs must not contain personal data.
URLs	URLs must not contain session identifiers.
notifications	Error situations must include error codes, and the user must be shown the error code together with the error message.
notifications	Error messages must be logged.
identity_systems	The application must not create a new identity system; it must rely on existing national (eID) or OS-based identity systems (e.g., Kerberos).
identity_systems	Apply address data system requirements.
identity_systems	Apply classifier system requirements.
identity_systems	Phone numbers must be standardized.

authentication	Authentication solutions for external users must comply with the national authentication requirements document.
logout	System logout must be explicit, understandable, and secure.
database	Applications using databases must have at least two database users.
database	Foreign keys must be used when referencing tables; all foreign keys must be indexed and formally defined.
database	All database tables must define an integer or UUID surrogate primary key; real-world data fields must not be used as primary keys.
database	Database field lengths must be defined in characters, not bytes, in DDL.
database	Database object names must be in English, semantic, and may contain only Latin1 letters (a–z, A–Z), digits (0–9), and underscore (_); names must not start with digits.
database	The application must be partitioning-agnostic.
logging	Application logging must use standard tools allowing administrators to configure output (file, database, syslog), log level, and format.
logging	All components must log events in structured JSON format compliant with OpenTelemetry (OTel).
logging	Logs must be in English.
build	Packaging, deployment, updates, rollback, and test execution must be automated using a standard widely used tool.
build	The built package name must include the project name and version and contain only permitted characters.
build	Applications must be built and deployed only from a uniquely referenced branch of the source repository.

build	Code must be delivered to the TRAM infrastructure repository; GitHub delivery is allowed only if mirrored internally.
build	Java libraries must be included as part of the application package.
build	Fonts, stylesheets, and JavaScript files must be served from the application itself.
build	Dependency versions must be fixed (pinned).
deployment	The application must be independent of a specific application server.
deployment	Version upgrades must be reversible.
deployment	The application must not assume the security of its environment.
deployment	Application servers must be dynamically addable to and removable from a service cluster.
deployment	All references to files or external systems must be externally configurable and not hardcoded.
deployment	Database or schema installation must not require exceptional user privileges.
information_security	Unless otherwise specified, the application must support systems complying with ISKE class K2T2S2 (or equivalent E-ITS security level).
information_security	The system must not expose internal operational information (e.g., full file paths, stack traces) to users.
information_security	All public network traffic must be encrypted using TLS; TLS parameters must be controlled by administrators, not developers.
information_security	Apply OWASP Top 10 mitigation measures based on application risk analysis.
information_security	Session cookies must be protected.
information_security	Implement CSRF protection.

information_security	Validate inputs on both frontend and backend.
information_security	Implement Content Security Policy (CSP).
information_security	All inter-component communication must be encrypted using TLS with appropriate certificates; configuration guidance must be included in installation documentation.
information_security	Apply the principle of minimal network access.
cryptography	Cryptographic algorithms must comply with RIA guidance.
cryptography	Cryptographic algorithms must be replaceable with minimal changes.
cryptography	Cryptographic keys must be securely managed.
data_protection	Applications must ensure compliance with personal data protection requirements.
data_protection	The application must not use external user-activity analytics services outside TRAM infrastructure (e.g., Google Analytics).
availability	High availability capability is required; each deployable component must be installable in multiple instances unless otherwise agreed.
availability	For high-availability systems, session management must be defined at the beginning of the project; shared file systems (e.g., NFS) must not be used for session storage.
monitoring	Each deployable component must provide machine-readable monitoring data (e.g., heartbeat.json), including name, version, external system status, last startup time, packaging time, and server time.

Appendix 3– Non-exclusive license for reproduction and publication of a graduation thesis[12]

I, Urve Aavik

1 grant Tallinn University of Technology free license (non-exclusive license) for my thesis “Principles and Application of Secure Development in the Public Sector”, supervised by Toomas Lepik,

1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2 I am aware that the author also retains the rights specified in clause 1 of the non-exclusive license.

3 I confirm that granting the non-exclusive license does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

[12] The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.