



A quantitative security evaluation and analysis model for web applications based on OWASP application security verification standard

Shao-Fang Wen^{*}, Basel Katt

Department of Information Security and Communication Technology, Norwegian University of Science and Technology, Gjøvik, Norway



ARTICLE INFO

Keywords:
Web application security
Security evaluation
Quantitative approach
Security analysis

ABSTRACT

In today's digital world, web applications are popular tools used by businesses. As more and more applications are deployed on the web, they are seen as increasingly attractive targets by malicious actors eager to exploit any security gaps present. Organizations are always at risk for potential vulnerabilities in their web-based software systems, which can lead to data loss, service interruption, and lack of trust. Therefore, organizations need to have an effective and efficient method for assessing and analyzing the security of acquired web-based software to ensure adequate confidence in its use. Quantitative security evaluation employs mathematical and computational techniques to express the security level that a system reaches. This research focuses on improving the quantitative analysis of web application security evaluation. We strive to unite the Open Web Application Security Project's (OWASP) Application Security Verification Standard (ASVS) into a structural and analyzable model, which aims to efficiently evaluate web application security levels while providing meaningful insights into their strengths and weaknesses.

1. Introduction

Web applications (or web apps) have been the mainstream technology for providing information and services over the Internet. Numerous businesses from various sectors continue to move their operations online. Most institutions and organizations employ web applications such as blogs, social networks, webmail, banks, and others that provide essential operational activities while also storing sensitive data. The widespread adoption of web applications in modern society has also captured the attention of hackers who are intent on taking advantage of any vulnerabilities in these applications to perpetrate malicious acts, resulting in the inefficiency and ineffectiveness of business activities (Erşahin and Erşahin, 2022). With the prevalence and significance of web applications nowadays, organizations want confidence that the software is developed securely and reliably that takes care of the required security mechanisms while minimizing the risks to the assets.

To gain the necessary confidence in acquiring and maintaining software systems, organizations need a thorough method for evaluating and analyzing the security of the software (Erşahin and Erşahin, 2022). Security evaluation seeks to provide trustworthy results for decision-makers to utilize (Gritzalis et al., 2002). The process involves evaluating the adequacy of security controls and procedures to address

them as well as identifying and analyzing security threats, vulnerabilities, and risks (Herrmann, 2002). To ensure stakeholders can make the most of this data, it must be presented in a suitable format for their needs. Quantitative security evaluation is a specialized field where computational and mathematical techniques are used to evaluate the level of security in a system (Gritzalis et al., 2002; LeMay et al., 2011). It seeks to access more precisely how much effort is required to defend the system or how high the danger of the system being compromised (Vache, 2009). Such a kind of security assessment model leads to clearly measurable security scores, giving a clear indication of the strength of a system's protection measures (Gritzalis et al., 2002).

The advantages of quantitative security evaluation approaches are obvious, however, the research work in this area seems limited. According to a recent survey on web application security (Ruan and Yan, 2018), there is not yet a practical and effective model for evaluating the security of web applications, while a majority of the research conducted has only looked at the typical security vulnerabilities associated with the application. Consequently, much must be done to develop an efficient model for assessing web security (Ruan and Yan, 2018). Not only that, in a recent systematic literature review research, Shukla et al. (Shukla et al., 2021) concluded that the majority of security assurance and evaluation research has been focused on qualitative perspectives, with

* Corresponding author.

E-mail address: shao-fang.wen@ntnu.no (S.-F. Wen).

Table 1
Examples of ASVS requirements.

Section	Requirement Description	L1	L2	L3	CWE	NIST 800-63
2.1 Password Security	2.1.1 Verify that user-set passwords are at least 12 characters in length (after multiple spaces are combined).	V	V	V	521	5.1.1.2
	2.1.3 Verify that password truncation is not performed.	V	V	V	521	5.1.1.2
	2.1.4 Verify that any printable Unicode characters, including language-neutral characters such as spaces and Emojis, are permitted in passwords.	V	V	V	521	5.1.1.2
	2.1.5 Verify users can change their password.	V	V	V	620	5.1.1.2
	2.1.11 Verify that "paste" functionality, browser password helpers, and external password managers are permitted.	V	V	V	521	5.1.1.2

minimal effort devoted to developing quantitative methodologies.

To complement the research gap, this paper focuses on the modeling of web application security evaluation that is more amenable to quantitative analysis. Specifically, we strive to synthesize Open Web Application Security Project's (OWASP) Application Security Verification Standard (ASVS) (OWASP 2022) and integrate it into a structural and analyzable model. OWASP ASVS is widely used for web application security assessment and security requirements elicitation, as it provides a comprehensive overview of all security-related topics. Despite having advantages for conducting security assessments, the operational nature of ASVS makes it challenging to generate meaningful data for analysis. Our quantitative approach enables the conversion of ASVS data into informative, understandable information. By combining meaningful data sets with sound analytics, security stakeholders can make informed choices that drive organizational decision-making (Wen et al., 2022). In this paper, we also demonstrate and illustrate how such models are used for the analysis of security strengths and weaknesses as well as quantitative aspects through aggregating ASVS verification results.

The rest of this paper is organized as follows. Section 2 presents the scientific background. In Section 3, we provide an overview of related work. In Section 4, we discuss the concept of system boundaries, which forms the foundation of our proposed model. In Section 5 the proposed security evaluation model is discussed in detail. Subsequently, Section 6 provides an example of data analytics based on this model to better illustrate it. Lastly, the conclusion and future works are presented in Section 7.

2. Scientific background

In this section, we provide the scientific background for our research

by introducing OWASP ASVS and offering compelling examples that demonstrate some security assurance analysis scenarios utilizing ASVS. By intertwining theory with practical applications, this section highlights the gap between conceptual understanding and real-world implementation, reinforcing the significance of ASVS as a valuable resource for ensuring analyzability.

2.1. OWASP application security verification standard

The OWASP is a non-profit, community-driven organization that promotes software security through educational materials, open-source software, and other initiatives. The OWASP ASVS is an open standard for performing web application security verification, which is designed to methodically test application and environment-level technical security controls. With this, it is possible to identify various potential vulnerabilities, for example, Cross-Site Scripting (XSS) and SQL injection. The ASVS Project has designed its standard for practical, "commercially workable". With extensive coverage and flexibility, the ASVS can be applied in various situations, from intimate internal security measuring to instructing developers how to suitably implement safety functions or evaluating third-party software and contractual development agreements. The latest stable version of ASVS is 4.0.3 released in October 2021.

The ASVS contains 286 verification requirements that are grouped into 14 higher-level categories (named "Chapter") and sub-categories (named "Section") that are of similar functionality. Additionally, from version 4.0, ASVS provides a comprehensive mapping to the Common Weakness Enumeration (CWE) (MITRE 2023). The Common Weakness Enumeration (CWE) is a list of weaknesses in software that can lead to security issues. While the CWE list is long, it is also prioritized by severity of risk, providing organizations and developers with a good idea about how to best secure applications. Where applicable, ASVS requirements are also mapped to (or aligned with) different security standards, including OWASP Proactive Control (OWASP 2023) and the U.S. National Institute of Standards and Technology (NIST) Digital Identity Guidelines (NIST 800-63) (Grassi et al., 2017) (See Table 1). The former describes the most important control categories that every architect and developer should follow, while the latter introduces modern, evidence-based, and advanced authentication controls. Fig. 1 depicted the whole data structure of ASVS.

2.2. Motivating examples of security assurance analysis

There is no doubt that the OWASP ASVS is of great utility when it comes to conducting web application security assessments based on the predefined list of specifications (Harrison et al., 2016; Sönmez, 2019). This initiative makes it possible for organizations to carry out a comprehensive security review and quickly get the risk posture. While such a security control 'checklist' can provide an output corresponding to the requirements of the standard, it is insufficient for organizations that need to make security decisions. In this section, we provide a motivating example, demonstrating some analysis scenarios using ASVS.

Let us assume that a security assessment is conducted on an open-

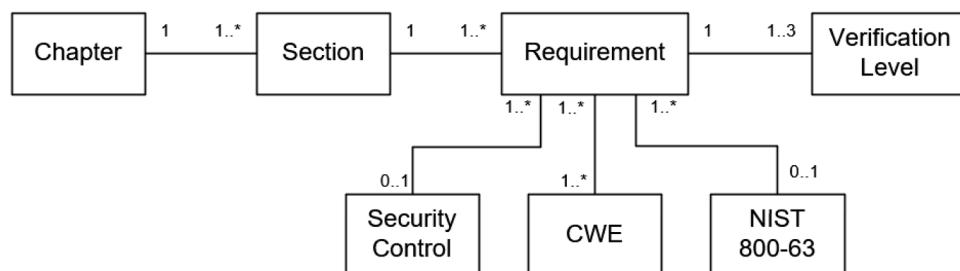


Fig. 1. ASVS data structure.

Table 2

An example of an ASVS assessment result.

Chapter	Pass	Fail	Score
Architecture, Design, and Threat Modeling	38	8	38
Authentication	27	8	27
Session Management	16	3	16
Access Control	5	2	5
Validation, Sanitization, and Encoding	24	2	24
Stored Cryptography	13	1	13
Configuration	48	2	48

Table 3

Exemplary security mechanisms with associated ASVS requirements.

Security Mechanism	ASVS Requirement
Password strength policy	V2.1.1-Verify that the user-set passwords are at least 8 characters in length (after multiple spaces are combined). V2.1.2-Verify that passwords of at least 64 characters are permitted and that passwords of more than 128 characters are denied. V2.1.4-Verify that any printable Unicode character, including language-neutral characters such as spaces and Emojis, are permitted in passwords. V2.1.7-Verify that passwords submitted during account registration or password change are checked against an available set of, at least, the top 3000 passwords. V2.1.9-Verify that there are no password composition rules limiting the type of characters permitted. There should be no requirement for an upper or lower case or numbers or special characters. V2.1.10-Verify that the application does not require periodic credential rotation. V2.3.1-Verify system-generated initial passwords or activation codes SHOULD be securely randomly generated, SHOULD be at least 6 characters long, MAY contain letters and numbers, and expire after a short period. These initial secrets must not be permitted to become the long-term password.
Password input functionality	V2.1.11-Verify that "paste" functionality, browser password helpers, and external password managers are permitted.
Password changing functionality	V2.1.5-Verify users can change their password. V2.1.6-Verify that password change functionality requires the user's current and new password.
Password processing logic	V2.1.3-Verify that passwords are not truncated.

source web application, which is installed and hosted on a virtual machine in the organization's intranet. The assessment result for each ASVS requirement is given either 0 or 1, where value 0 means the requirement is "Fail", and 1 means "Pass". The analysis task would be, first, to derive scores for *chapters* and *sections* using an aggregation algorithm, for example, computing the sum of the associated requirements. Table 2 shows an example of such datasets, which permit scores to be analyzed from the levels of *chapters*. Although the dataset provides a categorical view of security scores, it does not come up with a broader perspective and more strategic point of view. Instead of immediately delving into the complex relationship between process and technology, a common analytical approach is, to begin with a top-down analysis (B. Garrette et al., 2018; Sabatier, 1986). This involves examining the broader macro aspects first, from which a governing thought is arrived. The phrase "governing thought" implies the identification of a central or overarching idea that can guide the subsequent analysis (B. Garrette et al., 2018). This governing thought serves as a foundational framework for further investigation and interpretation. Once such macro aspects have been examined, a key principle or concept emerges that plays a crucial role in shaping the understanding of the subject matter. Thus, in this assessment context, to recognize discrepancies between the "runtime/operational environment" and the "software" itself, it is important to first separate the assessment scores of two entities. Likewise, the

scores of "process" related requirements (involving development and operation phases) should be distinguished from the above. Such 'aspects' provide a tidy collection of the stakeholders' views of the areas of interest in security evaluation (which will be elaborated on later).

Another deficiency we identify in ASVS is the lack of capability of system diagnosis for subject-of-matter at a granular level. A granular-level diagnosis goes beyond surface-level assessments and delves into a more comprehensive and meticulous analysis of various facets of the system (Delen and Ram, 2018; Pröllochs and Feuerriegel, 2020). It involves scrutinizing system components, configurations, and processes in detail, leaving no stone unturned in the pursuit of identifying potential issues and vulnerabilities. For example, say an analyst wants to make further analysis of password security; to identify "concrete" security mechanisms that need improvement or development in this category. In this regard, rather than analyzing scattered descriptive statements, it is more effective and efficient to use concise items to represent the associated requirements. Table 3 provides examples of mechanisms for password security, as well as the applicable requirements found in the ASVS. To enable a deeper level of security evaluation, we suggest that security requirements should be organized into a synthesizable and analyzable format.

In addition to analyzing the positive side of system security, organizations should strive for a comprehensive view of their system security posture by assessing possible structural flaws and weaknesses. ASVS provides consolidated mapping to CWE; however, additional information is required to answer crucial analysis questions, for example, what are the threats (or risks) that the weakness could result in? To what extent does the found weakness impact the security properties of the system? The ASVS framework offers a comprehensive set of measures for assessing the security of web applications. However, raw data obtained from ASVS alone is relatively challenging to process and analyze. Our modeling approach is to bridge the gap by facilitating the transformation of ASVS operational data into insightful, easy-to-analyze information, from which security stakeholders can then generate actionable knowledge, to be encouraged a better understanding of the existing context.

3. Related work

There is a wealth of research on security assurance and evaluation methods. Over the years, numerous frameworks and standards have been developed to analyze security. Common Criteria (CC) (Herrmann, 2002) is one of the most well-known efforts in this area. CC is an international ISO/IEC 15,408 standard for the security evaluation of IT products. The standard outlines a clear set of guidelines and specifications that provide organizations with the necessary information to accurately specify their security functional requirements and security assurance requirements. In addition, the CC offers a strict, standardized, and repeatable methodology to ensure the safe implementation, evaluation, and operation of a product at a suitable security level as prescribed by the operational environment. Furthermore, while comprehensive in scope, using this standard results in detailed documentation that often requires substantial effort when assessing products or services against a specific CC assurance grade (Ekclhart et al., 2007; Zhou and Ramacciotti, 2011). In addition, there are several security maturity models available for the software security domain, such as the Building Security In Maturity Model (BSIMM) (McGraw et al., 2009) and OWASP Software Assurance Maturity Model (OpenSAMM) (OWASP 2022). BSIMM is a research initiative that investigated the various approaches to software security employed by businesses, leading to the development of a framework featuring 116 activities and 12 practices. Like BSIMM, openSAMM is an open software security framework developed by OWASP, which provides guidelines on which software security practices should be used and how to assess them. Such maturity models provide frameworks, especially in a qualitative fashion, to evaluate the security posture of the process and culture practiced in an

organization.

Although various research studies have been conducted on web application security evaluation, few attempts have been made to establish a generic approach that quantifies the results systematically. Below are several papers that discuss this research area. The authors in (Hai and Nga, 2018) presented a security evaluation framework for web-portal security assessment, which integrates ISO/IEC 15,408 (ISO/IEC 2023) and OWASP evaluation model Common Criteria Web Application Security Scoring (CCWAPSS) (Charpentier, 2023). This framework facilitates numerical rankings via the use of a scoring system to assess the significance of each factor within the criteria. By doing so, it provides practical security evaluations that web portal developers can quickly understand and implement. Okamura et al. (Okamura et al., 2013) discussed a quantitative security evaluation approach for software systems from the vendor's viewpoint, centering on the analysis of collectible vulnerability data. They apply a stochastic model using a non-homogeneous Poisson process to explain this data, and then use numerical examples to evaluate the security measures relative to the content management system of an open-source project. Yautsiukhin et al. (Yautsiukhin et al., 2008) introduced a method of computing the security qualities of software architectures with the adoption of security patterns. The core metric used in this evaluation was threat coverage, and an algorithm was proposed to aggregate low-level measures associated with these patterns into a single high-level indicator. Lastly, Banaei and Khorsandi (Banaei and Khorsandi, 2012) presented a hierarchical structure for web service security, complete with a model that evaluates various aspects of security from an analytical perspective. They use the Analytical Hierarchy Process (AHP) theory to prioritize weighted averaging of critical security properties, such as authorization, confidentiality, and availability — all to provide greater levels of customization in terms of provider/consumer needs.

Furthermore, alternative methods for quantitative security assurance of IT systems have been proposed by some researchers. These concepts could be applied in software systems/web applications. For instance, Katt and Prasher (Weldehawaryat and Katt, 2018) outlined a quantification method to evaluate the security assurance of systems. This framework measures two parts: (1) the confidence that existing mechanisms are sufficient to meet security requirements; and (2) which potential security threats might leave a system vulnerable. The framework has been validated through case studies on public REST APIs. Ouedraogo et al. (Ouedraogo et al., 2009) utilized quantitative risk measurement techniques to create indicators that can be used to assess IT infrastructure security, alongside aggregation procedures. The primary algorithms used to perform operational aggregation are the recursive minimum, maximum, and weighted sum algorithms. Each of these tools has been designed to take into consideration a wide range of datasets when consolidating information. Pham and Riguidel (Pham and Riguidel, 2007) introduced an aggregational method that can be applied in the calculation of the security assurance value of the whole system when combining several entities, which have been evaluated independently. The effects of the emergent relations are taken into account in the calculation of the security assurance value of an attribute in the context of a system.

4. Model from the boundary of software system

As a web-based application can incorporate various resources and multiple environmental elements, it is crucial to take a wide-ranging approach to view the system, which is often not clear. In this section, we will first provide an overview of the idea of system boundaries for available software systems. This will supply a groundwork that can be utilized for our proposed model, and also show its pertinence to the field of security evaluation.

In general, a software system is comprised of elements that have been purposefully incorporated into the environment. These include the software structure and environmental factors around which we can

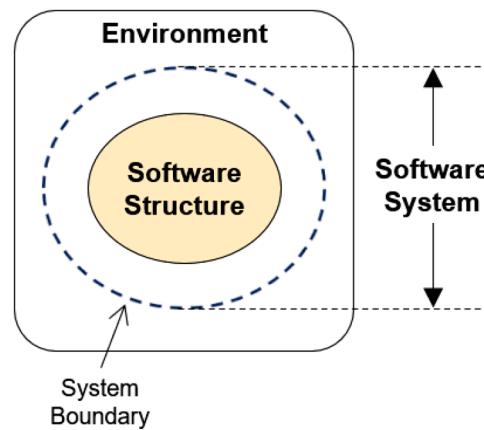


Fig. 2. The system boundary of software systems.

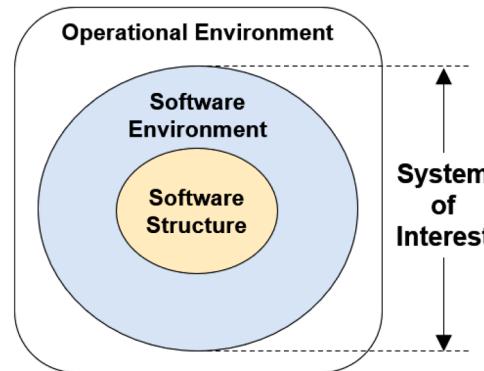


Fig. 3. The system boundary of the system of interest.

reasonably draw a boundary (i.e., the system boundary), as depicted in Fig. 2. The *software structure* is the core subset of the software system, meaning any source code or object code made to perform a specific task (s). An *environment* is a set of factors (e.g., facilities, operating conditions, or influences) that are available to a software component when it is being installed, executed, or operated. It is useful to think of an environment as being made up of things that are not part of the software component but can affect the software system's behavior.

From the perspective of a security evaluation, the system boundary defines what should be analyzed within the *System-of-Interest* (SOI), and thus distinguishes it from the external environment. Our security evaluation model divides this external environment into two categories: software environment and operational environment. This classification is made by distinguishing a line that serves as a conceptual boundary between them. The scope of SOI is comprised of the software environment and the system structure, whereas the operational environment is external to the system (depicted in Fig. 3). In our definition, a software environment refers to the complete set of hardware and software (tools, resources, systems, and services) that are the necessities to secure build, maintain, and scale the software components. Simple examples of software environments are a hardware environment, a software-based execution environment, or some combination of these. On the other hand, the operational environment contains elements and further systems that interact in some way with the software system, for example, a user, a system administrator, an organization, a LAN, or a general office environment. The operational environment might also be incorrectly implemented and managed and consequently contain errors that would result in flaws, however, following the definition of the system boundary, no assessment is made regarding the correctness of the operational environment. In terms of a security evaluation, it is assumed that the operational environment is absolutely precise and will aid the software

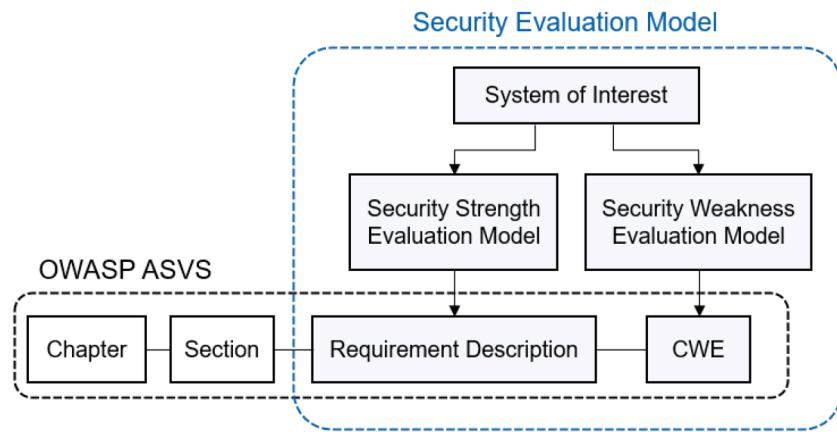


Fig. 4. The conceptual framework of the modeling approach for security evaluation.

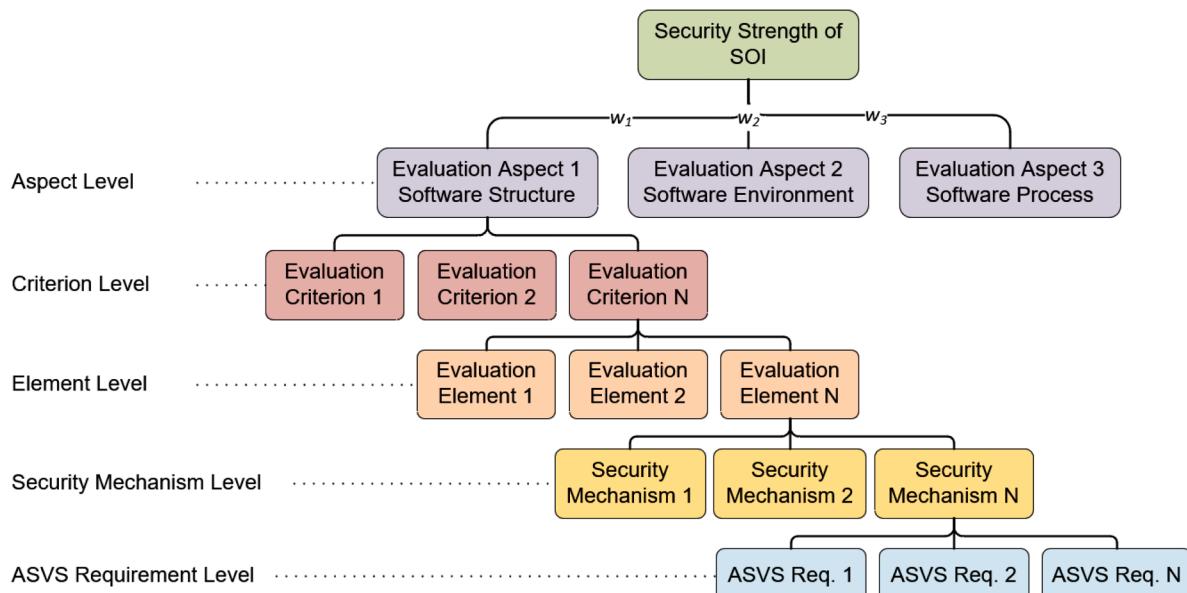


Fig. 5. Illustration of the layers that make up the hierarchical approach for the security-strength evaluation.

system in delivering its features accurately and securely.

5. Creating a structured and analyzable model

For a more complete security evaluation and analysis, it is essential to take into account both the advantages and disadvantages of the system's security. Our strategy is to make ASVS quantifiable by breaking it down into two core components: security strength evaluation and security vulnerability evaluation. The aim is to gain measurable insights that will aid in deepening our comprehension of the ASVS verification result. As shown in Fig. 4, our modeling approach is built upon the ASVS framework. The security strength evaluation measures system security by analyzing and gauging relevant security requirements. Additionally, the assessment of security weaknesses brings understanding to the overall system's risk posture—including how any uncovered weaknesses (i.e., CWE) impact the elements' security characteristics and their potential interactions with threats. At the simplest level, the security-strengths model can provide a measure of assurance that the system will be able to withstand attack, while the weaknesses model can identify the potential consequences when the security mechanisms are not properly implemented. In the following sections, we describe our approach to modeling the evaluation component of the system of

interest, how the security mechanisms can be represented structurally, and how the weakside of system security can be derived from ASVS.

5.1. Security strength evaluation model

The security strength of a system is defined as its security state, which reflects its readiness for security measures to defend against potential threats (Schechter, 2004). As shown in Fig. 5, we evaluate system security strength through a structural hierarchy of five levels to gain insight into its security capability. We begin by categorizing the strength assessment into three aspects: structure, environment, and process. Each evaluation aspect includes a two-level categorization method to classify the security mechanisms connected with the ASVS requirements.

The process of "evaluation" utilizes mathematical algorithms to assign numeric values to each component (i.e., evaluation components). In our approach, the scores of evaluation components are computed using a bottom-up approach, which involves the estimation of the lowest possible level in the model. Each ASVS requirement is assigned a numerical score and the scores are aggregated to create an overall score. Score aggregation is beneficial as it reduces the subjective bias in evaluating claims and provides a more objective method for determining the accuracy of claims (Andrews et al., 2006). The evaluation task begins by

determining the scores of ASVS requirements and aggregates their values by using an Average scheme to rate a set of evaluation components along the hierarchy. Since the security-strength evaluation hierarchy is designed in such a way that lower-level nodes (successors) will cover a part of the one node at the higher level, the components we add together are similar ones. Under this condition, using "Average" can consider all the relevant items so that we can derive a representative score of the whole data set. Finally, the overall score of the SOI is calculated using a weighted average scheme using the scores of evaluation scores along with their assigned weighting factors. This single value provides an objective measure of the system's security level. The used notation and the detailed evaluation process are discussed in the following.

5.1.1. Evaluation of ASVS requirements

The first step of assessment is responsible for the evaluation of the ASVS verification requirements. Initially, each ASVS verification requirement is mapped to one *verification case* to determine its fulfillment. As such, a score for each verification requirement could be determined, based on the observation of the verification case. Results for verification cases are quantified as 1, 0, and 0.5, depending on the level of fulfillment. The score 1 is given to the cases that pass the verification, indicating the corresponding requirements are fully fulfilled, while 0 means the requirements are not fulfilled (i.e., the verification case failed). A score of 0.5 implies the requirement is considered a partial fulfillment. Partial fulfillment means that the actual result matches its expected result, however, there might encounter unnecessary (or superfluous) exceptions/messages that are caught during the test-case execution. Such a test execution state is usually applied in the context of manual testing, heavily reliant on the tester's judgment (Reddy).

At this step, we use $S(ASVS_i)$ to denote the score of the i^{th} ASVS requirement, which can be expressed in Eq. (1).

$$S(ASVS_i) \in \{0, 1, 0.5\} \quad (1)$$

5.1.2. Evaluation of security mechanisms

It is possible to consider the requirements and mechanisms to be synonyms because they are frequently used in an abstract context. In the security evaluation and analysis approach, we attempt to use a more fine-grained "Security Mechanism" than descriptive ASVS requirements. Security mechanisms can be treated as the fundamental means and methods that are designed to achieve security-relevant purposes. The capabilities and behaviors provided by security mechanisms are specified in security requirements. That is, a security mechanism is an output of the implementation of (a set of) security requirements. While security requirements (in ASVS) are designed for verification, security mechanisms, on the other hand, are for analysis purposes. To provide analyzability, the mechanism must be small and simple enough to be evaluated. The exemplary security mechanisms can be found in Table 3.

To calculate the scores of security mechanisms, let $C(SecurityMechanism_i)$ denotes a set of ASVS scores associated with the i^{th} security mechanism, defined by Eq. (2).

$$C(SecurityMechanism_i) = \{S(ASVS_j) \rightarrow SecurityMechanism_i\} \quad (2)$$

We use $S(SecurityMechanism)$ to represent a measurement to reflect the actual (calculated) score of the security mechanism. The following formula (Eq. (3)) represents the calculation of the i^{th} security mechanism, which uses the average function to derive the score.

$$S(SecurityMechanism_i) = \frac{\sum C(SecurityMechanism_i)}{|C(SecurityMechanism_i)|} \quad (3)$$

5.1.3. Evaluation of criterion and element levels

The term "criteria" as used in this model refers to a higher, more abstract level of meaning that can be thought of as a standard in the SOI's application domain. These criteria are part of the "target" that the

Table 4
Corresponding evaluation criteria for each evaluation aspect.

Evaluation Aspect	Evaluation Criteria
Software Structure	Authentication Access Control Input Validation and Output Encoding Session Management Cryptography Error Handling and Logging Privacy and Data Protection Communication Intrusion Detection and Prevention Business Logic Security Files and Resources Security Memory Management Web Service and API Security
Software Environment	Environment Management Communication Hardening Configuration Hardening General Practice
Software Process	Security Requirement Secure Design Secure Coding Secure Code Review Secure Build and Deployment

Table 5
Evaluation elements in evaluation criteria.

Evaluation Criteria	Evaluation Element
Authentication	Authentication Architecture Password Security Authenticator Security Credential Storage Credential Update Credential Recovery Multi. Factor Authentication Authentication Logging Service Authentication
Access Control	Access Control Architecture Operation Level Access Control HTTP Request Access Control Access Control Logging

work is planned to achieve. These criteria are selected, tested, and measured to confirm the sufficiency of system security to be offered to users. Table 4 lists the corresponding evaluation criteria for each evaluation aspect. Evaluation criteria are then narrated in detail by a set of evaluation elements. Some examples of evaluation elements are presented in Table 5.

Similar to the algorithm in the previous level, the score of the i^{th} element-level component, denoted by $S(Element_i)$ is calculated using Eq. (4).

$$S(Element_i) = \frac{\sum C(Element_i)}{|C(Element_i)|} \quad (4)$$

where:

$$C(Element_i) = \{S(ASVS_j) \rightarrow Element_i\}$$

Consequently, the formula for calculating the score of the i^{th} criterion-level components is as Eq. (5).

$$S(Criterion_i) = \frac{\sum C(Criterion_i)}{|C(Criterion_i)|} \quad (5)$$

where:

$$C(Criterion_i) = \{S(Element_j) \rightarrow Criterion_i\}$$

5.1.4. Evaluation of aspect level

Aspects are the viewpoints about how stakeholders can describe the security strength at the highest level. While ASVS categorical analysis provides stakeholders with a detailed state of the security criteria, the aspect will allow them to see which particular viewpoint the system performs in a positive, neutral, or negative way. Therefore, they can dive deeper into details to get a complete picture of the dedicated aspect. While defining the aspects, we first include the predominant attributes of SOI, that is, *software structure* and *software environment*, mentioned in Section 3. The evaluation of the software structure aims to access the sufficiency of the ‘technical’ security mechanisms of the software system itself, including security architectures and security functionalities. The evaluation criteria under the software structure are, for example, authentication, access control, and cryptography. The evaluation of the software environment entails an examination of the environmental factors that contribute to the production and maintenance of the software system. organizational and physical facilities (for example, development, production, delivery, and operation) are among these factors.

In addition to the security aspect described above, developing and maintaining secure systems rely on the processes linking people and technologies (Kim and Solomon, 2010). Therefore, a secure software system should also provide evidence that it is developed and operated using adequate *software processes*, and conformance to implementation standards, including secure coding standards, adequate testing, verification and validation, and suitable specification and documentation for all system aspects. The ASVS also sets out security requirements related to the software process, for example, “V1.1.7-Verify availability of a secure coding checklist, security requirements, guideline, or policy to all developers and testers”. In certain circumstances, however, evaluating the software process is unfeasible, such as in open-source software development contexts. To work around this limitation, stakeholders must consider whether to factor in the aspect of the software process when conducting their security evaluation.

Following the previous score calculation pattern, the formula for calculating the score of the i^{th} aspect-level components is defined as Eq. (6).

$$S(\text{Aspect}_i) = \frac{\sum C(\text{Aspect}_i)}{|C(\text{Aspect}_i)|} \quad (6)$$

where:

$$C(\text{Aspect}_i) = \{S(\text{Criterion}_j) \rightarrow \text{Aspect}_i\}$$

5.1.5. Evaluation of SOI

So far, we have defined the measurement for the individual evaluation aspect, then we can derive a measure of the overall score for the SOI, with a simple reduction of a set of scores to a single ‘figure-of-merit’. The higher the value, the better the trustworthiness of the security strength. The aggregation function used in calculating the score of SOI is the weighted average, an approach that is highly intuitive and comprehensible. Rather than each assurance aspect contributing equally to the result, the use of a weighing factor emphasizes the contribution of particular assurance aspects over others to the security evaluation result, thereby highlighting those aspects in comparison to others in the analysis. The stakeholder can decide which of the three aspects is either more or less important to him (or her), given what the software is functioning for the organization. For example, the stakeholder could assign the following weighting factors 0.6, 0.3, and 0.1 to the aspects of software systems, software environments, and software processes respectively, to account for the proportion of corresponding importance. The exact weighting values are calculated or assigned based on the opinions of stakeholders by applying decision-making techniques that must be carried out based on the verification context. Finally, we standardize the score by scaling the value in the range of [0, 10].

Table 6
Security level.

Score	Security level
[0.0 – 1.0)	No Security
[1.0 – 4.0)	Low Security
[4.0 – 7.0)	Moderate Security
[7.0 – 9.0)	Good Security
[9.0 – 10.0]	Excellent Security

The overall assurance score of the SOI, represented by $S(\text{SOI})$ is calculated b Eq. (7).

$$S(\text{SOI}) = \sum_{i=1}^3 S(\text{Aspect}_i) \times w_i \times 10 \quad (7)$$

where:

w_i : the weight that corresponds to the i^{th} evaluation aspect ($0 < w_i < 1$ and $\sum w_i = 1$)

To offer better comprehensibility, we apply a discrete rating scheme in the final score of SOI. Table 6 is adapted from the NVD Vulnerability Severity Ratings (W3C 2022). In NVD, the higher score represents greater severity. However, our table shows the opposite definition, i.e., levels of security. With this table, we can be used to convert the score to a textual form.

5.2. Security weakness evaluation model

From the analysis view, for security weakness evaluation the most concerned subject is the consequence of the security weakness. The security weakness evaluation aims to describe the consequence of the found weakness in the SOI. This involves defining the taxonomy of effects, including security risks, potential threats, and the impact scopes (i.e., the violated security properties), covering the relationships among them. The comprehensive mapping to CWE in ASVS allows us to derive a set of (negative) components, resulting from the weakness (depicted in Fig. 6), including impact scopes (i.e., the violated properties), technical impacts, threats, and security risks. By analysis of CWE, we could design the foundations of mentioned components. Therefore, the core idea of the modeling approach is that we can use existing CWE databases in combination with the well-known threat and vulnerability analysis methodologies to generate threat and security risk catalogs for each ASVS element. In the following, we explain each component and the derivation rules.

5.2.1. Evaluation of CWE

In security weakness evaluation, we use the CWE identities to calculate scores for security weakness components. This calculation is conducted through a summation function, which accounts for every element in the ASVS verification. We focus on assessing the severity of weaknesses in the SOI and measuring its significance based on these values.

Let $CWE\ i$ denote a CWE ID that existed in the CWE repository. Eq. (8) defines the set of ASVS scores mapped to a given CWE.

$$C(CWE\ i) = \{S(\text{ASVS}_j) \rightarrow CWE\ i\} \quad (8)$$

For each ASVS with a score of 0 (i.e., not fulfilled requirements), the corresponding CWE is assigned the value 1. The total score for CWE i is calculated by accumulating the ASVS score with Eq. (9).

$$S(CWE\ i) = \sum_{j \in C(CWE\ i)} e_j \quad (9)$$

where:

$$e_j = \begin{cases} 1, & \text{if } S(\text{ASVS}_j) = 0, \\ 0, & \text{otherwise.} \end{cases}$$

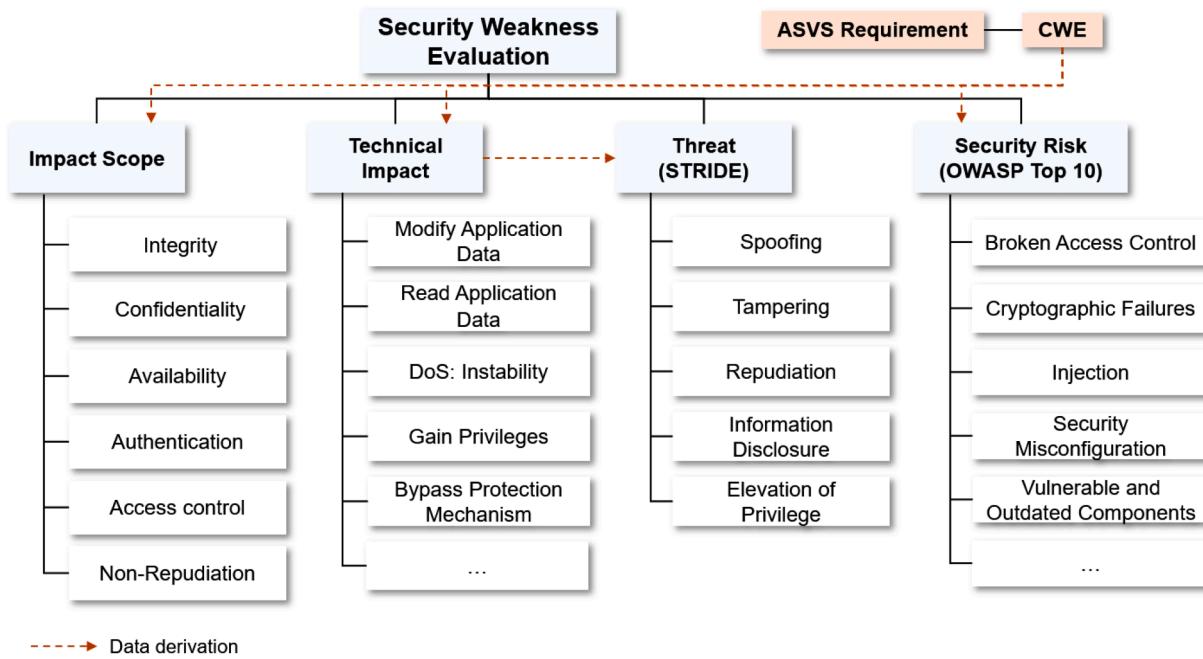


Fig. 6. Security Weakness Evaluation Model.

5.2.2. Evaluation of impact scope

Each SOI comprises the main security objective that needs to be achieved, like confidentiality and availability, which are commonly captured by a *security property*. For every security property, the impact must be evaluated. The *Impact Scope* element identifies the security property that is violated due to the existence of the weakness. As for the evaluation component, the impact scope is used to evaluate the severity of weakness with generic/abstract security requirements for the SOI. In the CWE model, the impact scope can be found in the attributes of *Common Consequences*. For example, the weakness “CWE-116: Improper Encoding or Escaping Output” impacts the security properties of Integrity, Confidentiality, Availability, and Access Control. Other impact scopes defined in CWE are Authentication, Authorization, and Non-repudiation.

To determine the score of the impact scope, we add up the corresponding CWE scores with Eq. (10).

$$S(\text{ImpacScope}_i) = \sum C(\text{ImpactScore}_i) \quad (10)$$

where:

$$C(\text{ImpacScope}_i) = \{S(\text{CWE } j) \rightarrow \text{ImpacScore}_i\}$$

5.2.3. Evaluation of technical impact

Technical Impact is the potential result that can be produced by the weakness, assuming that the weakness can be successfully reached and exploited. This is expressed in terms that are more fine-grained than confidentiality, integrity, and availability. The technical impact is an important criterion that can be useful to any organization that needs reasonable security assurance for their software-based solutions. The CWE ‘Common Consequence’ also describes the *Technical Impact* that arises if an adversary succeeds in exploiting this weakness. Security weaknesses can cause a lot of damage if they are successfully exploited. This information then evaluates the different types of damage that can be caused, and how severe the damage can be. Examples of technical impact are: Modify Data, Read Data, Unreliable Execution, Resource Consumption and Execute unauthorised Commands.

Similar to “Impact Scope”, the “Technical Impact” score is yielded by summing the results of the relevant CWEs using Eq. (11).

Table 7
Mapping of STRIDE categories based on CWE “Technical Impact”.

STRIDE Category	CWE/Technical Impact
Spoofing	Gain Privileges or Assume the identity
Tampering	Modify Application Data, Modify Memory, Modify Files or Directories, Unexpected State, Alter Execution Logic
Repudiation	Hide Activities
Information Disclosure	Read Application Data, Read Memory, Read Files or Directories,
Denial of Service	DoS: Instability, DoS: Resource Consumption (CPU), DoS: Resource Consumption (Memory), DoS: Crash or Exit or Restart, DoS: Resource Consumption (Other)
Elevation of Privilege	Execute unauthorised Code or Commands, Bypass Protection Mechanism

$$S(\text{TechnicalImpact}_i) = \sum C(\text{TechnicalImpact}_i) \quad (11)$$

where:

$$C(\text{TechnicalImpact}_i) = \{S(\text{CWE } j) \rightarrow \text{TechnicalImpact}_i\}$$

5.2.4. Evaluation of threat

To have a clear picture of the dangers, it is important to formulate an assessment of the threats to the SOI. Threat assessment is often performed on a higher level, especially addressing legal or business-related issues. In our test-based approach, threats are identified and evaluated based on the catalogs of known CWEs, deriving from the relevant verification results of ASVS. CWE with its Common Consequences provides a point where we could start. In terms of threat categories, we use the STRIDE framework (Shostack, 2014), which is a mature and optimal approach, to classify threats in areas where mistakes are often made. The acronym “STRIDE” stands for the threat categories of Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege.

CWE-521: Weak Password Requirements

Weakness ID: 521
Abstraction: Base
Structure: Simple

View customized information: **Conceptual** **Operational** **Mapping-Friendly** **Complete**

- ▶ **Description**
- ▶ **Extended Description**
- ▶ **Relationships**
- ▶ **Common Consequences**
- ▼ **Memberships**

Nature	Type	ID	Name
MemberOf	C	724	OWASP Top Ten 2004 Category A3 - Broken Authentication and Session Management
MemberOf	V	884	CWE Cross-section
MemberOf	C	951	SFP Secondary Cluster: Insecure Authentication Policy
MemberOf	C	1353	OWASP Top Ten 2021 Category A07:2021 - Identification and Authentication Failures

Fig. 7. An example of navigating to OWASP Top 10 in CWE (CWE-521).

The CWE Schema offers an alternative method for mapping between the CWE and the STRIDE, mediated by the attribute of “Technical impact”. We map CWE in the dataset against STRIDE using the “Technical Impact” attribute elicited from the previously mapped CWE. Each STRIDE category had a relationship with one or more enumerations of the Technical Impact. The mapping of STRIDE to CWE Technical Impact is presented in Table 7.

Based on the mapping table, threat scores are calculated using Eq. (12).

$$S(Threat_i) = \sum C(Threat_i) \quad (12)$$

where:

$$C(Threat_i) = \{S(TechnicalImpact_j) \rightarrow Threat_i\}$$

5.2.5. Evaluation of security risk

While the mapped CWE list in ASVS is extensive, it can be grouped and ranked by risk severity. The OWASP Top 10 categories provide an easy, clear at-a-glance summary of the ten most critical application vulnerabilities, which are arranged according to their impact and the security risk involved. The condensing of the numerous kinds of CWS into a small number of categories gives an easier way to analyze the security weakness in the software system. Instead of making an effort to eradicate all vulnerabilities, one can decide which of the ten risks is either more or less hazardous to the organization. This provides analyzers with a good idea about how to draw stakeholders’ attention to certain issues that are the most common problems at the time.

In our model, “Security risk” is derived from the “Memberships” attribute in CWE. Fig. 7 represents an example where the security-risk category (A07-Identification and Authentication Failures) is mapped for CWE-521: Weak Password Requirements. The evaluation of security risks involves the quantification of risks and the associated *Criticality* factor. When security risks are identified, it is difficult to remove all of them simultaneously due to the limited resources available for vulnerability mitigation. Criticality is a numerical value that we give to a security risk that communicates how serious it is and determines the mitigation to be applied first. The higher the criticality, the more urgent the need to act. A common criticality assessment method is based on the probability of failure and consequences. Criticality can be calculated using the following equation:

Table 9
Examples of non-applicable verification cases.

Criteria	Element	Security Mechanism	ASVS Requirement
Authentication	Authenticator Security	Look-up secrete security	V2.6.1-V2.6.3
		Out-of-band verifier security	V2.7.1-V2.7.6
		OTP verifier security	V2.8.1-V2.8.7
Input Validation and Sanitization	Input Validation	Input validation for LDAP Query	V5.3.7
		XPath query parameterization	V5.3.10
Privacy and Data Protection	Server-side Data Protection	Health Data Encryption	V6.1.2
		Financial Data Encryption	V6.1.3
Web Service and API Security	SOAP Web Service Security	Add integrity check to SOAP payload	V13.3.2
	GraphQL	GraphQL logic	V13.4.1-V13.4.2

$$\text{Criticality} = \text{Probability} \times \text{Severity}.$$

The equation to derive the score of a security risk is defined as Eq. (13).

$$S(SecurityRisk_i) = \sum C(CWE_j) \times c_i = \sum C(CWE_j) \times p_i \times s_i \quad (13)$$

where:

- c_i : the criticality that corresponds to the i th *SecurityRisk*
- p_i : the probability that corresponds to the i th *SecurityRisk*
- s_i : the severity that corresponds to the i th *SecurityRisk*

To evaluate the criticality, we refer to the data factors listed for each of the OWASP Top 10 categories (OWASP 2022), which are systematically derived using CVSS v3. Two data factors are considered: “Average Incidence Rate” and “Average Weighed Impact”. The former represents the *Probability* while the latter is the *Severity*. Table 8 shows the snapshot data factors of the one security risk “Broken Access Control” in OWASP Top 10. According to the table, the criticality factor of the security risk is

Table 8
Data factors of “Broken Access Control” in OWASP Top 10.

Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage	Total Occurrences
55.97 %	3.81 %	6.92	5.93	94.55 %	47.72 %	318,487

Table 10
Summary of evaluation-aspect scores.

Score of SOI	Security Level	Evaluation Aspect	Weight	Score
7.721	Good Security	Software Structure	0.6	0.45
		Software Environment	0.3	0.26
		Software Process	0.1	0.06

calculated as $3.18\% \times 5.93 = 0.225$.

6. Case study

To show the effectiveness of the suggested methodology, a manual security evaluation and analysis for a single web application has been performed. It's important to note that due to the technologies used in the software, some mechanisms were not put in place that would meet ASVS requirements. For example, requirements of V2.6.1 to V2.6.3 in the

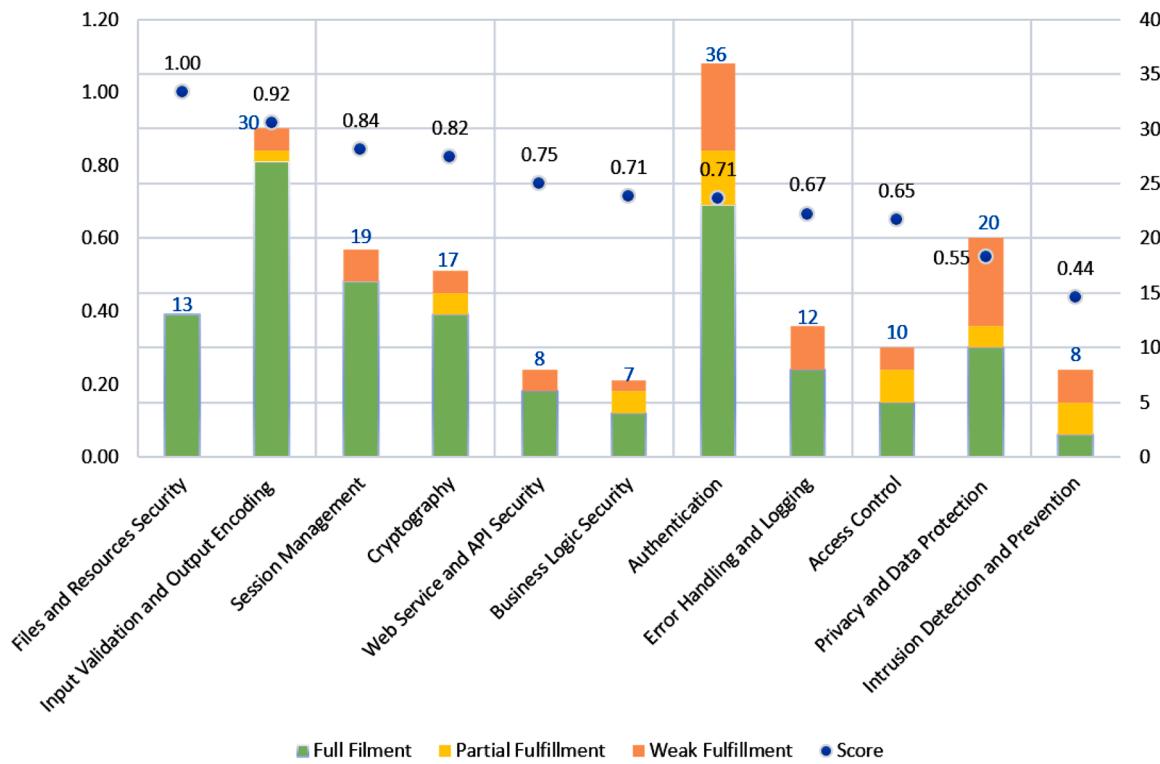


Fig. 8. Analysis of evaluation criteria scores.

Criteria	Score
Authentication	0.71
Access Control	0.65
Input Validation and Output Encoding	0.92
Session Management	0.84
Cryptography	0.82
Intrusion Detection and Prevention	0.44
Error Handling and Logging	0.67
Privacy and Data Protection	0.55
Business Logic Security	0.71
Files and Resources Security	1.00
Web Service and API Security	0.75

Element	Score
Authentication Architecture	0.75
Authenticator Security	0.60
Password Security	0.68
Credential Recovery	1.00
Credential Storage	0.70
Credential Update	0.25
Service Authentication	1.00
Authentication Logging	0.00

Security Mechanism	Score
Password Strength	0.64
Password Input Functionality	0.00
Password Changing Functionality	1.00
Password Processing Logic	1.00

Security Mechanism	Score
Notification Functionality of Credential Update	0.25

Element	Score
Data Protection Architecture	0.75
Server-Side Data Protection	0.71
Client-Side Data Protection	1.00
Sensitive Data Transmission	1.00
Data Backup and Retention	0.00
User-Managed Data	0.25
Sensitive Data Logging	0.00

Security Mechanism	Score
Cached Data Protection	0.00
Memory Data Protection	1.00
Sensitive Data Encryption	1.00
Data Collection Consent	1.00
Limited Access to Privacy-Related Features	1.00

Fig. 9. Drill-down analysis based on the security strength evaluation model.

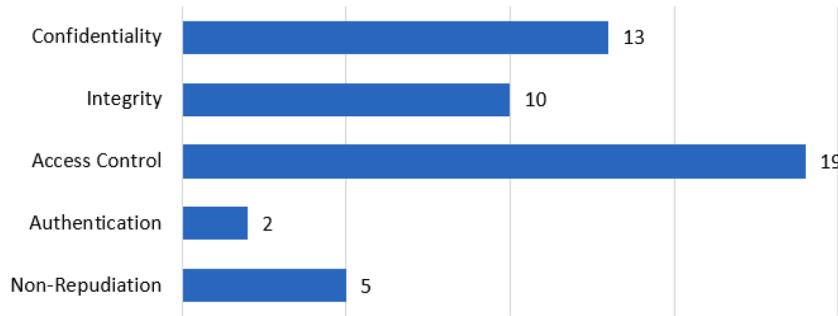


Fig. 10. Analysis of Impact Scopes.

Authentication criteria define the security mechanism of "Look-up secrete security". However, the application does not feature the specified functionality. Therefore, the relevant requirements may be excluded from the verification scope. As a result of this, these requirements were marked as "Not Applicable." Examples of non-applicable ASVS requirements in this case study are listed in Table 9. In summary, there are 261 out of 286 ASVS requirements have been determined to be "applicable" to the security verification.

6.1. Analysis results using the model

In this section, we describe analysis which is used mainly to support security analysts in formulating analytics for discovering, interpreting, and communicating significant patterns in data. We begin by calculating the security strength using the evaluation model and aggregating the verification findings of ASVS. Table 10 presents the summary of the SOI and evaluation-aspect scores. The SOI score is 7.721, which indicates that the SOI has a "Good Security" rating. The weight factors for the three evaluation aspects are given using a subjective weighting approach. In this case, the stakeholders rated the higher weight on "Software Structure" among the three aspects.

Fig. 8 depicts an example of the "next-level" security strength analysis, concentrating on the aspect of the software structure, in which the evaluation criterion scores are shown alongside the distribution of verification-case fulfillment. Among the 11 evaluation criteria, "Files and Resource Security" has the highest score (1.00) while "Intrusion Detection and Prevention" is the lowest (0.438). "Authentication" has the greatest number of verification cases and gains a moderate score of 0.708.

The built-in hierarchical structure in the security strength model allows for a very thorough breakdown and makes it simple to determine whether the necessary security mechanisms are implemented and

Table 12
Summary of threat scores with corresponding technical impacts.

Threat	Score	Technical Impact	Score
Spoofing	12	Gain Privileges or Assume Identity	12
Tampering	15	Modify Application Data	7
		Modify Memory	0
		Modify Files or Directories	4
		Unexpected State	3
		Alter Execution Logic	1
Repudiation	4	Hide Activities	4
Information Disclosure	17	Read Application Data	12
		Read Memory	0
		Read Files or Directories	5
Denial of Service	7	DoS: Instability	0
		DoS: Resource Consumption (CPU)	3
		DoS: Resource Consumption (Memory)	1
		DoS: Crash, Exit, or Restart	1
		DoS: Resource Consumption (Other)	2
Elevation of Privilege	11	Execute unauthorised Code or Command	1
		Bypass Protection Mechanism	10

functioning properly. Fig. 9 illustrates the drill-down scenarios in the security strength analysis. For instance, it is discovered that the security mechanism is deficient in "Notification Functionality of Credential Update" after looking into the low-scoring "Credential Update" (score = 0.25). Additionally, the evaluation's findings indicate that the "Password Input Functionality" (score = 0) doesn't seem to be prepared for "Password Security." Furthermore, in the criteria of "Privacy and Data Protection", "Cache Data Protection" is the only one of the five crucial security mechanisms in "Sever-Side Data Protection" that does not meet the requirements.

An analysis of the effect of the found CWE on the security properties

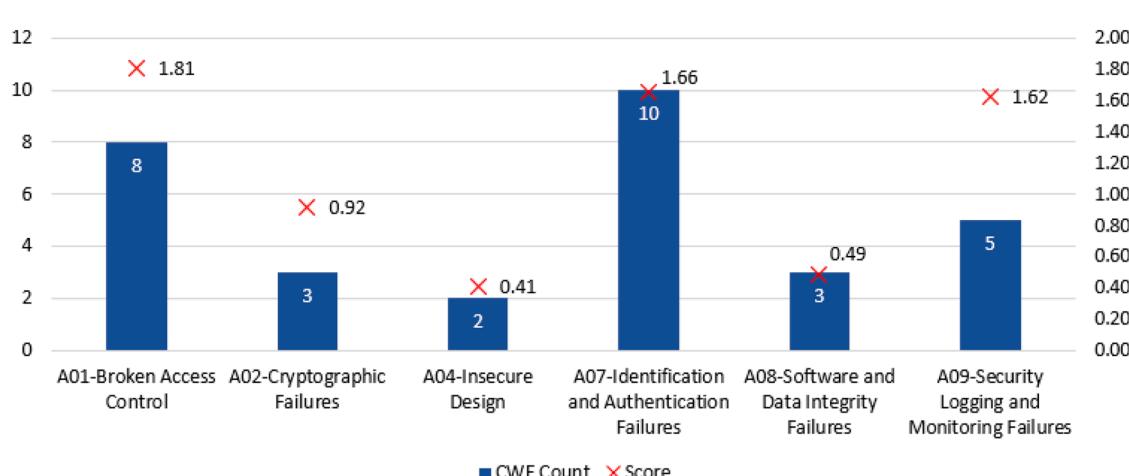


Fig. 11. Analysis of security risk.

Table 11
Summary of security risks.

Security Risk	Number of CWEs	Criticality	Score	Rank
A01-Broken Access Control	8	0.23	1.81	1
A02-Cryptographic Failures	3	0.31	0.92	4
A04-Insecure Design	2	0.20	0.41	6
A07-Identification and Authentication Failures	10	0.17	1.66	2
A08-Software and Data Integrity Failures	3	0.16	0.49	5
A09-Security Logging and Monitoring Failures	5	0.32	1.62	3

(i.e., the impact scope) is carried out by plotting it into a bar chart. Fig. 10 shows the analysis of impact scopes, where the horizontal axis denotes the numbers of CWE. In contrast to the positive connotation of security strength scores, in security weakness evaluation, the higher the score, the more serious a weakness/threat/security risk is. That means the scores of all weakness components always result in a negative effect on the result. Thus, it is clear from the figure that the system's flaws have the greatest impact on the security properties of "Access Control" and "Confidentiality".

We also assessed how CWEs result in OWASP's Top 10 security risks, which are summarized in Table 11 and depicted in Fig. 11. The results of our evaluation show that six of the ten critical risks are associated with SOI. The six risks were then ranked in order of their scores. As shown in the table, it is observed that even though the numbers of CWEs in "Identification and Authentication Failure" are the highest (10), the criticality rating is relatively low (0.17). Consequently, this risk is placed second according to the calculated score (1.66). Among the six risks, "Broken Access Control" is the most critical one with a score of 1.81, while "Insecure Design" is identified as least critical.

Finally, threat scores are produced using Eq. (12). The analysis of threats is shown in Table 12, highlighting the severity of threats applicable to the system. Out of the six threats, "Information Disclosure" is evaluated as the most serious. "Information Disclosure" is considered to be the relatively serious threat out of the six listed, with which the most significant technical impact is "Read Application Data".

7. Conclusion

This paper presents a model to quantify the security evaluation of web applications. By using methods such as aggregation, a gathering of secure scores, and analytics, organizations can better understand the security posture of the system of interest and make an informed decision. Our method allows for the incorporation of ASVS operational data into knowledge-based information, which can provide insight into the security strength of a given system and any potential vulnerabilities or threats. This approach to security evaluation ensures that the ASVS requirements are adequately considered, while also providing a comprehensive view of system security from the macro aspects of "structure", "environment", and "process". From there, the problem is broken down, identifying the components at the next level down, with a special focus on the critical or vital granular level of security mechanisms. Security mechanisms are the concise objects that provide a bridge between the descriptive ASVS requirements and their eventual analysis. In addition, in our test-based approach, we integrate the ASVS into the evaluation of discovered vulnerabilities (i.e., mapped CWE). This modeling approach applies existing CWE databases with effective mapping techniques to yield threat and risk catalogs that correspond with each ASVS element. This approach is achieved by taking the verification result as explicit input to the evaluation, which allows for a more precise and focused evaluation of any potential negative effect and thus enhances the overall analysis results.

It is important to recognize the limitations of this work to inform

further investigations. First, this model primarily focuses on technical security mechanisms and does not take into account human factors, such as social engineering attacks or insider threats. Additionally, the model does not guide how to mitigate risks, as it is meant to be used as an analysis tool and not a prescriptive guide. Overall, while the model is a useful tool for analyzing security posture, it should be used in conjunction with other frameworks and considerations to build a comprehensive security strategy. A future endeavor is to develop more practical security metrics based on the model and incorporate them into a security analytics application, as recommended in (Wen et al., 2022). This application serves to bolster data analysis, granting organizations the ability to thoroughly measure key elements of security. With its help, they can get a clearer picture of what needs to be done to ensure security and compliance. As an additional step, automation of the security evaluation process may be considered to enhance its efficacy and allow for real-time monitoring. This marks an important development in continuously refining the system's security measures by providing well-structured metrics and analytics.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Shao-Fang Wen reports financial support was provided by SFI-Norwegian center for Cybersecurity in Critical Sectors.

Data availability

No data was used for the research described in the article.

Funding

The Research Council of Norway financially supports this research work through the SFI-Norwegian center for Cybersecurity in Critical Sectors (NORCICS, NFR project number: 310105).

Acknowledgement

This research work is financially supported by the SFI Norwegian center for Cybersecurity in Critical Sectors (NORCICS, NFR project number: 310105).

Declaration of Generative AI and AI-assisted technologies in the writing process

Not applicable.

References

- Andrews, R., Boyne, G.A., Walker, R.M., 2006. Subjective and objective measures of organizational performance: An empirical exploration. *Public Service Performance: Perspect. Measure. Manage.* 14–34 pages.
- Banaei, O., Khorsandi, S., 2012. A new quantitative model for web service security. In: 2012 IEEE 14th International Conference on Communication Technology. IEEE.
- Charpentier, F., "Common Criteria Web Application Security Scoring CCWAPSS"; Available from: https://dl.packetstormsecurity.net/papers/web/ccwapss_1.1.pdf. (Accessed on Feb. 3, 2023).
- Delen, D., Ram, S., 2018. Research challenges and opportunities in business analytics. *J. Busi. Anal.* 1 (1), 2–12 pages.
- Eklhart, A., et al., 2007. Ontological mapping of common criteria's security assurance requirements. In: IFIP International Information Security Conference. Springer.
- Erşahin, B., Erşahin, M., 2022. Web application security. *South Florida J. Dev.* 3 (4), 4194–4203 pages.
- Garrette, B., et al., 2018. Sell the Solution: Core Message and Storyline. Cracked it! How to solve big problems and sell solutions like top strategy consultants 197–221 pages.
- Garrette, B., et al., 2018. Sell the solution: Recommendation report and delivery. Cracked it! How to solve big problems and sell solutions like top strategy consultants 223–249 pages.
- Grassi, P.A., Garcia, M.E., Fenton, J.L., 2017. Digital identity guidelines. *NIST Special Publication 800* pages 63–3.
- Gritzalis, D., Karyda, M., Gymnopoulos, L., 2002. Elaborating quantitative approaches for IT security evaluation. *Secur. Inform. Soc. Vis. Perspect.* 67–77 pages.

- Hai, H.D., Nga, P.T., 2018. Evaluating the security levels of the Web-Portals based on the standard ISO/IEC 15408. In: in Proceedings of the 9th International Symposium on Information and Communication Technology.
- Harrison, S., et al. 2016. "A security evaluation framework for UK e-government services agile software development". arXiv preprint.
- Herrmann, D.S., 2002. Using the Common Criteria For IT Security Evaluation. CRC Press volume.
- ISO/IEC, "Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 1: Introduction and general model"; Available from: <https://www.iso.org/standard/72891.html>. (Accessed on Jan. 21, 2023).
- Kim, D., Solomon, M.G., 2010. Fundamentals of Information Systems Security. Jones & Bartlett Publishers volume.
- LeMay, E., et al., 2011. Model-based security metrics using adversary view security evaluation (advise). In: 2011 Eighth International Conference on Quantitative Evaluation of SysTems. IEEE.
- McGraw, G., Chess, B., Miguez, S., 2009. Building security in maturity model. Fortify & Digital.
- MITRE, "Common Weakness Enumeration (CWE)"; Available from: <https://cwe.mitre.org/index.html>. (Accessed on Feb. 3, 2023).
- Okamura, H., M. Tokuzane, and T. Dohi. 2013. "Quantitative security evaluation for software system from vulnerability database".
- Ouedraogo, M., et al., 2009. Security assurance metrics and aggregation techniques for it systems. In: 2009 Fourth International Conference on Internet Monitoring and Protection. IEEE.
- OWASP, "OWASP Proactive Controls"; Available from: <https://owasp.org/www-project-proactive-controls/>. (Accessed on Feb. 3, 2023).
- OWASP, "OWASP Top10 Introduction"; Available from: https://owasp.org/Top10/A00_2021_Introduction/. (Accessed on Apr. 27, 2022).
- OWASP, "Software Assurance Maturity Model v2.0"; Available from: <https://www.opensamm.org/>. (Accessed on Apr. 30, 2022).
- OWASP, "Application Security Verification Standard (ASVS)"; Available from: <http://owasp.org/www-project-application-security-verification-standard/>. (Accessed on Jun. 3, 2022).
- Pham, N., Riguiedel, M., 2007. Security assurance aggregation for it infrastructures. In: 2007 Second International Conference on Systems and Networks Communications (ICSNC 2007). IEEE.
- Pröllochs, N., Feuerriegel, S., 2020. Business analytics for strategic management: Identifying and assessing corporate challenges via topic modeling. *Inform. Manage.* 57 (1), 103070 pages.
- Reddy, N. "An Excellent Compilation of Software Testing Concepts (Manual Testing)".
- Ruan, Y.L., Yan, X.Q., 2018. Research on key technology of web application security test platform. In: Proceedings of International Conference on Education, Management and Social Science (EMSS 2018) (2018).
- Sabatier, P.A., 1986. Top-down and bottom-up approaches to implementation research: a critical analysis and suggested synthesis. *J. Public Policy* 6 (1), 21–48 pages.
- Schechter, S.E., 2004. Computer Security Strength and risk: a Quantitative Approach. Harvard University.
- Shostack, A., 2014. Threat modeling: Designing for Security. John Wiley & Sons.
- Shukla, A., et al. 2021. "System Security Assurance: A Systematic Literature Review". arXiv preprint.
- Sönmez, F.Ö., 2019. Security qualitative metrics for open web application security project compliance. *Proced. Comp. Sci.* 151, 998–1003 issue, pages.
- Vache, G., 2009. Vulnerability analysis for a quantitative security evaluation. In: 2009 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE.
- W3C, "RDF 1.1 XML Syntax"; Available from: <https://www.w3.org/TR/rdf-syntax-grammar/>. (Accessed on Jan. 26, 2022).
- Weldehawaryat, G.K., Katt, B., 2018. Towards a quantitative approach for security assurance metrics. In: The 12th International Conference on Emerging Security Information.
- Wen, S.F., Shukla, A., Katt, B., 2022. Developing Security Assurance Metrics to Support Quantitative Security Assurance Evaluation. *J. Cybersecur. Priv.* 2 (3), 587–605 pages.
- Yautsiukhin, A., et al., 2008. Towards a quantitative assessment of security in software architectures. Nordic Workshop on Secure IT Systems (NordSec), Date: 2008/10/01–2008/10/01. Copenhagen, Denmark, Location.
- Zhou, C., Ramacciotti, S., 2011. Common criteria: Its limitations and advice on improvement. *Inform. Syst. Secur. Assoc. ISSA J.* 24–28 pages.
- Shao-Fang Wen** (Ph.D.) is currently working as a post-doctoral research fellow in in Norwegian University of Science and Technology. The areas of his research include
 - Socio-technical security analysis
 - Security education and learning
 - Software security and secure programming
 - Security assurance and security testing
 - Knowledge management and ontology
- Basel Katt** is currently working as a Professor in Norwegian University of Science and Technology. The areas of his research are:
 - Software security and vulnerability analysis
 - Security assurance and security testing
 - Model driven software development and model driven security
 - Access control, usage control and privacy protection
 - Security monitoring, policies, languages, models and enforcement