Routledge
Taylor & Francis Group

Journal of Applied Security Research

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/wasr20

Ontology-Based Metrics Computation for System Security Assurance Evaluation

Shao-Fang Wen & Basel Katt

To cite this article: Shao-Fang Wen & Basel Katt (2022): Ontology-Based Metrics Computation for System Security Assurance Evaluation, Journal of Applied Security Research, DOI: 10.1080/19361610.2022.2157190

To link to this article: https://doi.org/10.1080/19361610.2022.2157190

Published online: 19 Dec 2022.

JOURNAL OF APPLIED SECURITY RESEARCH
https://doi.org/10.1080/19361610.2022.2157190

Routledge
Taylor & Francis Group

Ontology-Based Metrics Computation for System Security Assurance Evaluation

Shao-Fang Wen and Basel Katt

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

ABSTRACT

Security assurance evaluation (SAE) is a technique that helps organizations to appraise the trust and confidence that a system can be operated correctly and securely. This paper contributes to the research on quantitative SAE by proposing an ontology-based assurance metrics computation solution, which consists of (1) a quantitative SAE approach, (2) an ontology for modeling the security assurance components and metrics, and (3) a metrics calculation engine for automatically generating metrics values. The feasibility and effectiveness of the proposed ontology-based SAE approach are examined through a preliminary ontology evaluation as well as a practical application-based evaluation.

KEYWORDS

System security; security assurance; quantitative approach; security metrics; ontology

Introduction

Nowadays Information and Communication Technology (ICT) systems have become increasingly ubiquitous in various domains of human societies, such as telecommunications, financial services, power supply, transportation, and more. Many organizations across all sectors are internally and externally connected by networks of ICT systems and heavily rely on such so-called cyber systems for conducting daily operations and achieving competitive advantage. Organizations always have a primary concern that there may be vulnerabilities existing in their ICT environments that can compromise organizational data, disrupt business services, and jeopardize trust. The loss of availability, integrity, and confidentiality in these systems and services can result in a harmful impact on organizations.

To mitigate the potential loss or the compromise of critical ICT systems, many organizations choose to either acquire systems that have built-in security mechanisms or implement them in later phases (Waddell et al.,

CONTACT Shao-Fang Wen shao-fang.wen@ntnu.no Department of Information Security and Communication Technology, Norwegian University of Science and Technology, Gjøvik, Norway.

2011). In any case, organizations need shreds of evidence that show the security mechanisms are correctly and effectively put in place and carry out their intended functions to prevent, detect or divert a risk or to reduce its impact on a system's assets (Ouedraogo et al., 2013). It is therefore important for organizations to know, on the one hand, if their assets are susceptible to threats, and on the other hand, if the security mechanisms can be effective in fulfill the requirements of mitigating threats (Weldehawaryat & Katt, 2018). To gain the necessary confidence in acquiring and maintaining ICT systems, organizations need a method for assessing the security of the system, including the impact the system may have on the organization's risk posture. Security assurance stands out as the way to address such a need.

According to Anderson (2020), assurance can be seen as "our estimate of the likelihood that a system will not fail in some particular way." Similarly, security assurance can be defined as our estimate that the system will not be compromised somehow. From a substantive perspective, security assurance refers to the degree of confidence that security needs are satisfied (Spears et al., 2013). A formal definition of security assurance can be found in the NIST Special Publication (Ross, 2011), which states "Security assurance is a critical aspect in determining the trustworthiness of information systems. It is the measure of confidence that the security features, practices, procedures, and architecture of an information system accurately mediates and enforces the security policy" (Page 26). In this paper, we follow the definition provided by Katt and Prasher (2019) and, thus, view security assurance "the confidence that a system meets its security requirements and is resilient against security vulnerabilities and failures." From the definition, we assume that the confidence indicated by the security assurance represents the level of trust in terms of security requirements that are fulfilled, as well as vulnerabilities and threats that are mitigated.

To address the issue of "confidence in security," for the past decades, there has been a greater emphasis on system security assurance evaluation (SAE), which results in various methods, processes, and models developed by researchers or practitioners (Shukla et al., 2021). SAE can be generally categorized into qualitative and quantitative methods. Qualitative methods are employed to study events or regulatory control of a Target of Evaluation (TOE), and understand the security posture of its implementation, while quantitative methods use computational techniques and mathematical data to express the security level of a TOE. The advantages of applying quantitative approaches in SAE are highlighted in the research work of Gritzalis et al. (2002), including (1) supporting decision-making by using and comparing metrics, (2) expressing ICT security with less complicated and more coherent mechanisms, (3) providing a basis to the security

status achieved by implementing different security controls, and (4) developing models with useful information about the behavior of TOE in different contexts. Utilizing metrics to capture and evaluate the security posture of ICT systems has gained attention in recent years. Such metrics are intended to deliberate the assurance aspect of the system security to reliably transfer information (Weldehawaryat & Katt, 2018; Katt & Prasher, 2018). The advantages of quantitative SAE approaches are obvious, however, the research work in this area seems limited. According to the findings of the systematic literature review, Shukla et al. (2021) concluded that a major of the SAE research has been focused on the qualitative perspective and very few efforts have been made toward developing quantitative security assurance methodologies. Consequently, scarce work has described how to combine quantitative SAE while taking into account both metrics modeling and computation.

This paper aims to complement the research gap by proposing a security assurance metrics calculation solution in the quantitative SAE, particularly using an ontology-based approach. Ontologies have been regarded as an effective approach to semantic-driven modeling and have been used in various research fields, such as education, information integration, and knowledge management (Raad & Cruz, 2015). The main benefit of the ontology-based model is the availability of a formal, encoded description of the domain knowledge: that is, all the concepts, their attributes, and their inter-relationships will be well-defined and represented (Berners-Lee et al., 2001). Unlike pure mathematics methods, the ontology-based hybrid approach can explicitly represent data semantics, and thus it can effectively describe the critical concepts of SAE. Due to the formalization, it can be represented and to some degree interpreted by machines and enables the formal analysis of the domain, which allows an automated or computer-aided extraction and aggregation of knowledge from different sources and possibly in different formats (Gruber, 1993).

The research work is built on the previous works on SAE presented by the authors Katt and Prasher (2019) and Wen et al. (2022) that adds particular semantic web technologies for advancing the contribution. The primary objective is to formulate an explicit security assurance ontology that can extend the quantitative SAE framework with the formal definition of assurance components (which will be explained later) and achieve automatic assurance metrics calculation. In this paper, we present the core model of the proposed SAE ontology as well as the design, implementation, and evaluation of the overall ontology-based approach. The rest of this paper is organized as follows. In section Related work, we provide an overview of related work. The quantitative SAE metamodel is introduced in

Section Quantitative SAE metamodel. Section Ontology-based approach explains the details of the proposed ontology-based approach while the evaluation is described in Section Evaluation. Lastly, the conclusion and future works are presented in Section Conclusion and future work.

Related work

In this section, we address related work in (1) quantitative approaches to security assurance, and (2) the application of ontologies in the domain of security assurance.

Quantitative security assurance evaluation

Research on security assurance and evaluation methods is vast. In the past, various frameworks and standards have been developed for evaluating security. One of the most representative works is Common Criteria (CC) (Herrmann, 2002). The CC is an international standard (ISO/IEC 15408) for the security evaluation of IT products. It provides a set of guidelines and specifications that can facilitate the specification of security functional requirements and security assurance requirements. With the strict, standardized, and repeatable methodology, the CC assures implementation, evaluation, and operation of a security product at a level that is commensurate with the operational environments. Despite being a standard, the drawback of such a comprehensive methodology is that the documentation is complicated and needs a large effort in preparation for the evaluation of a product or service against a specific CC assurance level (Ekclhart et al., 2007; Zhou & Ramacciotti, 2011). Some other examples of security maturity models are the Building Security In Maturity Model (BSIMM) (McGraw et al., 2009) and OWASP Software Assurance Maturity Model (OpenSAMM) (OWASP, 2017) and OWASP Application Security Verification Standard (ASVS) (OWASP, 2021a), which are provided for the software security domain. BSIMM is a study of how different organizations deal with software security, which resulted in a software security framework that is organized into 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. OWASP ASVS provides guidelines for web application security testing and corresponding security controls. It also lists a set of security assurance requirements and an associated qualitative evaluation scheme that consists of three maturity levels.

In the past, however, few efforts have been made to provide a generic approach to quantify the security posture to support SAE systematically. Several papers in this research area are highlighted below. Liu and Jin (2015) conducted a study to analyze the security threats and attacks on the WLAN network architecture and developed a security assessment and enhancement system. This system is divided into two subsystems, a security assessment system, and a security enhancement system. The security assessment system is based on fuzzy logic and analyzes the vulnerability of the physical layer (PHY) and medium access control (MAC) layer, key management layer, and identity authentication layer. This approach provides a quantitative value of security level based on security indexes. Whereas the security enhancement system is an integrated, trusted WLAN framework based on the trusted network connection that helps improve WLAN's security level. Agrawal et al. (2019) used the Fuzzy Analytical Hierarchy Process (Fuzzy-AHP) methodology to evaluate usable security. They also assessed the impact of security on usability and the impact of usability on security using a quantitative approach. Katt and Prasher (2019) proposed a general-purpose security assurance framework and its assurance evaluation process. The basic components of the proposed framework included are the security assurance scheme, security assurance target, security assurance metrics, security assurance technique, evaluation evidence, and assurance level. The framework and process depend on quantitative security assurance metrics that were developed too. They discussed the advantages of quantitative security assurance metrics considering both the security requirements and vulnerabilities.

Furthermore, several researchers have been working on SA metrics development and calculation. For instance, 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. Ouedraogo et al. (2009) take advantage of quantitative risk measurement methodologies to develop metrics for IT infrastructure security assurance evaluation along with aggregation techniques, i.e., the assurance level of a system is a specific combination of assurance levels from underlying components. The main algorithms used for the operational aggregation include the recursive minimum algorithm, the recursive maximum algorithm, and the recursive weighted sum algorithm. Moreover, to help businesses address service security assurance, Ouedraogo (2012) presents a set of metrics that can estimate the level of confidence for both consumers and providers. The

defined metrics can be categorized into three main areas: security-related metrics (existence, correctness, etc.,), security verification-related metrics (coverage of verification, depth of verification, etc.), and privacy-related metrics (data confidentiality and service consumer anonymity).

Some SA-metrics methodologies use evidence and arguments over security measures adequacy in a security case to build an acceptable level of confidence in system security. For instance, Rodes et al. (2014) propose the use of security arguments by facilitating security metrics that need to be complete and valid and propose a framework for argument assessment that generates and interprets security metrics on the example of software systems. Within the framework, security is quantified in terms of a level of beliefs, i.e., a confidence level of arguments. Several approaches take advantage of patterns to assess and evaluate system security. In this area, for instance, Heyman et al. (2008) associates security metrics with patterns, and exploit the relationships between security patterns and security objectives to enable the interpretation of measurements. Fernandez et al. (2010) evaluate the security of architecture by considering different misuse patterns. They propose to analyze how many misuse patterns for architecture can be countered when adding security patterns to improve the architecture. The calculated value then represents the level of security for the applied security patterns. Lastly, Villagrán-Velasco et al. (2020) evaluate system security based on threat enumeration and on verifying if these threats are not controlled in specific software architectures. They also consider the effect of policies and the use of weights according to their impact.

Ontologies in security assurance

In computer science, one of the most accepted definitions of ontologies is the one provided by Gruber (1993), who defines an ontology as "an explicit formal specification of a conceptualization." This is further elaborated that an ontology is a formal description of the concepts and relationships in an area of interest, simplifying and abstracting the view of the world for some purpose (Wand et al., 1999). Based on the ability to formalize domain knowledge and facilitate interoperability in knowledge sharing, ontology technology has recently played a vital role in cyber security domains (Fenz & Ekelhart, 2009; Tsoumas & Gritzalis, 2006). There is a large number of research papers related to the field of ontology-based security assurance and evaluation. Among them, we can highlight papers aimed at modeling security assurance methodologies, security assessment techniques, as well as ontology-based security metrics management.

Raskin et al. (2001) are one of the first to introduce ontological approaches to security evaluation. He implies that one of the ultimate goals

is the inclusion of semantic data sources to facilitate the formal specification of the information security community know-how for the support of routine and time-efficient measures to prevent and counteract computer attacks. Analyzing the current state of the art, a major portion of ontologies directly relate to the Common Criteria (CC) methodologies (Herrmann, 2002). The CC is an international standard (ISO/IEC 15408) for the security evaluation of IT products. With the strict, standardized, and repeatable methodology, the CC assures implementation, evaluation, and operation of a security product at a level that is commensurate with the operational environments. Despite being a standard, the drawback of such a comprehensive methodology is that the documentation is complicated and needs a large effort in preparation for the evaluation of a product or service against a specific CC assurance level (Ekclhart et al., 2007; Zhou & Ramacciotti, 2011). In this context, several research works focus on modeling CC knowledge and assurance specifications. Yavagal et al. (2005) present an ontological approach to the modeling of the CC security functional requirements. Through object-oriented modeling techniques, the ontology achieves categorization and classification of CC security requirements and related domain knowledge by creating hierarchical representations of CC requirements, producing a structure where the high-level requirements identified in the non-leaf nodes are decomposed into specific criteria in the leaf nodes. To conquer the time-consuming issue of CC assurance evaluation, Ekclhart et al. (2007) developed a CC Ontology, focusing on modeling the security requirements documented in CC. Based on ontology, a tool is created to support the CC evaluation process in several ways, for example, document preparation, linking, and tagging. Finally, Białas (2013) presents an ontology-based approach to the elaboration and management of the IT security evaluation process complaint with the CC standard. Based on the related knowledge base features, this ontology work focuses on the issues concerning evidence, which are elaborated for the given IT product or system according to the assurance level claimed for it, and later, are independently evaluated together with the target of evaluation.

Some research efforts have attempted to employ ontology-based techniques in security assessment or the security domain for integrating security assurance methodologies. Franco Rosa et al. (2018) propose a security assessment ontology, named SecAOnto, to conceptualize the main knowledge in the domain of security assessment, aiming to support security assessment methods based on assessment criteria. The core concepts included in SecAOnto can be generally categorized into (1) system assessment, (2) information security, and (3) security assessment. Wang and Guo (2009) propose an ontology-based approach to analyzing and assessing the security posture of software products. It provides quantitative

measurements for a software product based on an ontology built for vulnerability management, called OVM (Wang & Guo, 2009). Users can query the OVM to infer similar products and collect the related vulnerability information for each product. for generating an overall score, they propose an algorithm based on the CVSS metrics of the vulnerabilities inside the software. Gao et al. (2013) proposed an ontology-based framework for assessing the security of network and computer systems from the attacker's perspective. The proposed taxonomy consists of five dimensions, which include attack impact, attack vector, the target of attacks, vulnerability, and defense. Aman and Khan (2019) presented an ontology-based security model that aims to provide the necessary knowledge to evaluate the security performance of an application specifically in the context of its hosting infrastructure. The ontology consists of three domains: (1) The infrastructure domain contains the necessary vocabulary to set up a virtual operational environment for the target of evaluation, (2) the Testing profile domain is used to define, implement, and execute the test scope, requirements, and specifications, and (3) Security aptitude domain encompasses a list of all the known vulnerabilities that are assigned an impact score using the Common Vulnerability Scoring System (CVSS).

In addition, several ontologies for security assessment are proposed in certain domains or contexts. For example, in the domain of the Internet of Things (IoT), Gonzalez-Gil et al. (2019) proposed a context-based security evaluation ontology (IoTSecEv) to describe the different security preferences of the end-users of an IoT device, based on concerns and interests in different security elements, such as threats, vulnerabilities, security mechanisms, or features. In that regard, it is possible to evaluate security from a context-based standpoint in which the different interests and concerns of the uses are properly addressed. Moreover, in the context of the cloud system assessment, Koinig et al. (2015) established a knowledge-based ontology (Contrology) for capturing the knowledge required to audit a cloud computing environment. Their work is based on the ontology proposed by Fenz and Ekelhart (2009), which consists of six classes: assets, controls, security attributes, security recommendations, threats, and vulnerabilities. Likewise, Maroc and Zhang (2019) developed an ontology (CS-CASEOnto) for cloud services security evaluation, which covered necessary security knowledge and relevant cloud concepts of significance to security measurement. The ontology includes concepts related to the evaluation target, criteria, yardstick reflecting stakeholders' requirements, and the related evaluation activities including data gathering techniques and data synthesis approaches. In the automobile domain, Shaaban et al. (2019) presented an ontology (OnSecTa) for the security verification and validation process. The model verifies and validates security requirements in a vehicle to

assure that these requirements are fulfilled according to the security status, and the actual security goal needs to be achieved. It creates an ontological view of vehicle components and detected potential threats and related security requirements. With logical queries to the ontology, one can determine whether or not the security requirements can handle risks in a vehicle. In addition, Powley et al. (2019) proposed an Evaluation Ontology (EO) that facilitate the modeling of evaluation processes and outcome in automobile industries, specifically for connected vehicles. For dealing with the complex systems of systems that are vulnerable to cyberattacks, it aims to integrate different types of evaluation into a single model for all activities at all levels of all organizations in an enterprise.

Lastly, ontologies are also applied in security metrics management to foster quantitative security evaluation. Kotenko et al. (2013) proposed the ontology of security metrics which can be used for the tasks of security evaluation and countermeasure support in Security Information and Event Management (SIEM) systems. The evaluation is conducted using the information on discovered vulnerabilities or alerts on threats and attacks. Based on the value of the generating metric, a set of countermeasures will be provided for decision-making. Doynikova et al. (2020) propose a semantic model for the security evaluation of information systems in which an ontology of security metrics is developed to trace dependencies among available security knowledge sources, available raw security data, and metrics calculated on their base (divided by the security assessment goals) and security assessment goals. The key aspect of ontology is to process huge streams of gathered security-related data (both static, from open-source databases, and dynamic, from security monitoring tools).

Quantitative SAE metamodel

In this section, we present our quantitative SAE metamodel. With metamodels, we define different types of *information* and the corresponding *relationships* allowed in SAE modeling. To reach the reusability of the evaluation models as well as flexibility in assurance score calculation, the model must be sufficiently generic that it can be applied to any application domain, regardless of the subject of the evaluation. In this respect, our modeling approach for SAE follows a five-level hierarchical structure, in which each node represents a distinct *assurance component*, as shown in Figure 1. The assurance target is the product or system that is the subject under security evaluation, such as an information system, part of a system or product, or a cloud ecosystem. The evaluation serves to validate claims made about the assurance target. The core principle behind our proposal is that the SAE should be quantitative and distinguish two critical assurance

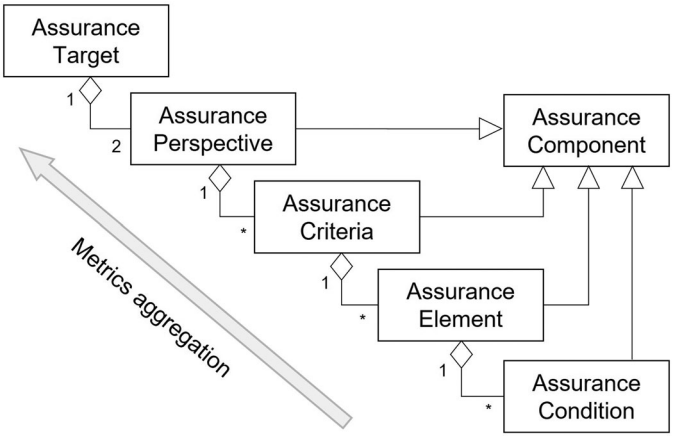


Figure 1. Compositions of the security assurance evaluation model.

perspectives: the protection side and the weakness side of the assurance target. Each perspective is composed of one or more criteria, and each criterion is composed of one or several elements until reaching the lowest level. Thereafter, this model considers the context of the intended environment (i.e., assurance conditions) in which the assurance target operates. Accordingly, security assurance scores of assurance components are computed using a bottom-up approach, which involves the estimation of the lowest possible level of detail. These estimates are then aggregated continuously in conjunction with predefined algorithms to arrive at a more fine-tuned final estimate at the top-level assurance target. For simplicity of presentation, we use "assurance" in short to represent the term "security assurance" in all component names. In the following sections, we will explain the details of the assurance components and how the assurance metrics are aggregated.

Assurance component

In our model, assurance components constitute the essential parts of assurance metrics calculation. The concepts for each component are described below.

Assurance perspective

Assurance perspectives describe the interrelation or relative significance in which an assurance target is evaluated. In our approach, two perspectives on cyber security are taken into evaluation: security requirements and vulnerabilities. The former addresses the positive side of system security while the latter considers the negative side. We assume that, on the one hand, fulfilling security requirements through implementing countermeasures and

checking its correct functionality will give protection against unintentional errors. On the other hand, checking the existence (or non-existence) of known potential vulnerabilities in the system gives the resistance assurance to intentional penetration or bypass. We argue that even if security mechanisms are properly elucidated at the requirement stage, they could result in weakness if they are inappropriately implemented or deployed. Consequently, while evaluating security assurance, security requirement improves the assurance posture; contrariwise, the existence of vulnerabilities leads to a reduction of the assurance level. Such concepts will be inherited by the rest of the assurance components.

Assurance criteria

Assurance criteria are the specific properties that will be selected, tested, and measured to confirm the sufficiency of system security to be offered to users. As used in this model, the term assurance criteria refer to a higher, more abstract level of meaning that can be thought of as a standard in the assurance target's application domain. These criteria are part of the "target" that the work is planned to achieve (or eliminate from the perspective of vulnerabilities). In our quantitative security assurance approach, assurance criteria play an especially important role in the assurance evaluation, which provides a basis for comparison among different assurance targets; a reference point against which another system can be evaluated. In Table 1 we give exemplary criterion sets for an assurance target in the domain of web applications, in which the content is extracted from the OWSAP ASVS (in defining security requirement criteria) and OWASP Top 10 (OWASP, 2021b; in defining vulnerability criteria). The former provides a rigorous list of security requirements for testers, developers, security professionals, and consumers, while the latter lists the ten most common web application security risks nowadays.

We argue that the foundation for quantifying and analyzing metrics in SA is to understand what "criteria" are of interest and of "how important" each is expected to be. The assurance criteria are formulated depending on the objectives and functions of the assurance target. Concerning security, not all security requirements should be treated equally important (Katt & Prasher, 2019). Likewise, the vulnerabilities in need of fixing must be prioritized based on which ones pose the most immediate danger. To reflect

Table 1. Exemplary assurance criteria in web applications.

Security requirement criteria	Vulnerability criteria
Authentication	Broken access control
Access control	Cryptographic failure
Validation, sanitization, and encoding	Injection
Data protection	Identification and authentication failure

that, one must specify a numeric factor for each assurance criterion: "Weight" for security requirement criteria and "Risk" for vulnerability criteria. On the one hand, the weighing factor emphasizes the contribution of particular aspects of security requirements over others to the security assurance result; thereby highlighting those aspects in comparison to others in the SA analysis. That is, rather than each security requirement (criteria) in the whole data set contributing equally to the result, some of the data is adjusted to make a greater contribution than others. The weight factor expresses how security is emphasized in the assurance target and it must be done based on the application context. For example, if authentication is necessary to make a specific API secure, that security requirement should be given particular importance, hence the weight is also high. On the other hand, from the perspective of vulnerabilities, the term risk can be defined as the probability and the consequence of an unwanted incident caused by existing vulnerabilities. That is, a risk is an impact of uncertainty on systems, organizations, etc. Several frameworks and methods have been developed for risk analysis, and organizations may choose their method depending on the type of risks they encounter, or their business area, for example, CVSS (Common Vulnerability Scoring System) (FIRST, 2012) and DREAD (Damage, Reproducibility, Exploitability, Affected Users, and Discoverability) (Burns, 2005).

Assurance element

Assurance criteria are narrated in detail by a set of assurance elements. Like in assurance criteria, assurance elements are divided into security requirement elements and vulnerability elements. The formal represents a requirement item needed to be fulfilled, while the latter indicates a particular kind of vulnerability potentially existing in the assurance target. Table 2 lists the exemplary elements with the corresponding assurance criteria.

Assurance condition

An assurance condition describes the underlying constraints (or terms) of assurance elements that need to be taken into consideration in SAE. It is specifically defined according to the organizational contexts, which include

Table 2. Exemplary assurance elements (security requirement elements).

Security requirement criteria	Security requirement element
Authentication	Password security
	Credential storage
	Credential recovery
	One time verifier
Validation, sanitization, and encoding	Input validation
	Sanitization and sandboxing
	Output encoding
	Deserialization prevention

Table 3. Exemplary assurance conditions (security requirement conditions).

Security requirement element	Security requirement condition
Password security	The passwords should be at least 64 characters are permitted, and passwords of more than 128 characters are denied. Password truncation is not performed. However, consecutive multiple spaces may be replaced by a single space. Any printable Unicode character, including language-neutral characters, such as spaces and Emojis, is permitted in passwords. Password change functionality requires the user's current and new passwords. A password strength meter is provided to help users set a stronger password.

special circumstance items, such as the deployment environment, the organization's current state, and security concerns. Besides, assurance conditions can be also represented as test cases performed to check to what extent the security requirements' conditions and the vulnerabilities' conditions are true. Table 3 represents the exemplary security requirement conditions under the element of "Password Security."

Assurance metrics computation

SAE is a systematic process of assigning meaningful scores to the assurance target that indicates its security posture (Ouedraogo et al., 2012). The higher the value, the better the trustworthiness of the system product against its security mechanisms. The overall evaluation of an assurance target is obtained from the scores of the assurance conditions and the criteria/elements of the evaluation model applied. We conceive the SAE as an aggregated value of multiple assurance conditions that is directly quantified from the test results. We suggest an aggregation method for doing so by using the model as the structure for estimating values related to SAE into a single measure. With this approach, SAE tends to be adaptive and quite accurate regarding the application domain and the organizational context. Also, assessing scores for test results of assurance conditions and aggregating these scores to the corresponding assurance element is easier than directly finding a single assurance element score.

Figure 2 depicts a sample hierarchical structure of the proposed SAE model, while Table 4 describes each notion. Our quantitative approach divides the SAE into three sequential phases: the first phase of evaluation is responsible for the assessment of the assurance elements; the second phase for the evaluation of the assurance criteria; and the third for the assurance perspectives, in turn, of the overall assurance level of the assurance target. With the term "evaluation" we refer to the assignment of a *metric* to each component in the model. Metrics represent measurement or evaluation

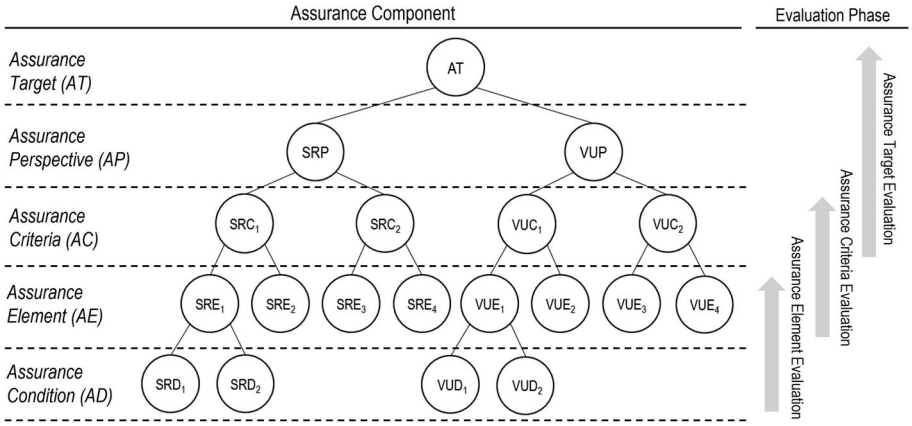


Figure 2. Sample hierarchical structure of the security assurance evaluation.

Table 4. Symbols used in the assurance evaluation.

Symbol	Description
AT	Assurance target
SRP	Security requirement perspective
VUP	Vulnerability perspective
SRC	Security requirement criteria
VUC	Vulnerability criteria
SRE	Security requirement element
VUE	Vulnerability element
SRD	Security requirement condition
VUD	Vulnerability condition

indexes that are given attributes to satisfy the SAE. The three-phase quantitative process is discussed in the following subsections.

Assurance element evaluation phase

The first phase of assurance evaluation is responsible for the assessment of the assurance elements of the SA evaluation model, from the quantification of the corresponding assurance conditions. In our test-based methodology, each assurance condition is mapped to one test case to decide fulfillment scores (for SRD) or existence scores (for VUD). For SRD, results for test cases are primarily pass or fail, where a pass indicates that the corresponding SRD is "Fully fulfilled" (fulfillment score = 1), while a failure of a test case means that the SRD is "Not fulfilled" (fulfillment score = 0). However, in some test cases, the result can be considered "Partially Fulfilled." Partial fulfillment means that: the actual result matches its expected result, however, there are more rigorous criteria/specifications needed to be met to strongly claim full fulfillment. In addition, an unnecessary (or superfluous) exception/message caught during the test-case execution can be also treated as a partial fulfillment (Arindaeng et al., 2018; Bosch et al., 2012). Such a test execution state is

usually applied in the context of manual testing and heavily relied on the tester's judgment (Reddy, n.d.). For example, it is assumed that the SHA-1 encryption algorithm is found in testing the SRD "The system stores account password in approved encrypted formats." In this case, even though there is evidence showing the password is encrypted, we see this test case to be a partial pass, as the SHA-1 is not considered a strong-enough password encryption function (Mirante & Cappos, 2013). Therefore, the assurance score of the SRD is assigned a value of 0.5, indicating "Partially fulfilled." Similarly, the existence score for VUD has two value options, where 0 means no vulnerability existence indicated by the test results, and 1 represents the existence of the vulnerability.

The scores for SRE and VUE are calculated separately. For an SRE, once the fulfillment scores are decided in all associated SRDs, its score can be calculated. We define a metric *ActSRD* as a measurement to reflect the actual (calculated) score of SRE. The value of *ActSRE* is obtained by averaging the fulfillment scores of the related SRD. Since the SRDs, we add together are similar ones, by using the "Average" function, we can consider all the relevant items to derive a representative score of the whole data set. Also, the assurance conditions are designed in such a way that each condition will cover one perspective of the assurance element. Failing the whole element if one condition fails is not fair for the rest of the conditions. The following formula represents the calculation of the *i*-th SRE score (represented as *ActSREi*):

$$ActSRE_i = \frac{\sum_{j=1}^{n} ActSRD_{ij}}{n}, \ \forall \text{ActSRD} \ \in .\{0, \ 05, \ 1 \qquad (1)$$

where,

ActSRDij: the actual (fulfillment) score of the j-th SRD associated with the i-th SRE;
n: the number of SRD associated with the i-th SRE

Similarly, the formula used for calculating the actual VUE score (*ActVUE*) is defined as the average of the corresponding VUD existence score, represented below:

$$ActVUE_i = \frac{\sum_{j=1}^{n} ActVUD_{ij}}{n}, \ \forall ActVUD_{ij} \ \in \ \{0, \ 1 \qquad (2)$$

where,

$ActVUD_{ij}$: the existence score of the *j*-th VUD associated with the *i*-th VUC;
n: the number of VUD associated with the *i*-th VU

Assurance criteria evaluation phase

The second phase of assurance evaluation is responsible for the calculation of assurance criteria scores. Based on the previous discussion, the actual score of the *i*-th SRC, represented by *ActSRCi*, is measured based on the average value of its respective SRE and obtained by multiplying a weight factor to express the levels of importance. The scale of the weight factor ranges from 1 to 10, where 1 is assigned to SRC that are least essential, while 10 is the maximum expressing a vital requirement. The formula to calculate *ActSRCi* is defined as:

$$ActSRC_i = WghSRC_i \times \frac{\sum_{j=1}^{n} ActSRE_{ij}}{n}, \ \forall WghSRC_i \ \in \ [1, \ 10] \qquad (3)$$

where,

ActSRCij: the actual score of the j-th SRE associated with the i-th SRE
WghSRCi: the weight factor that corresponds to the i-th SRC.
n: the number of SRE associated with the i-th SRC

Based on Equation (3), it can be derived that *ActSRC* has a maximum value, equaling its weight factor when all the underlining security requirements are fulfilled (i.e., $ActSREi = 1$).

The assurance metric *ActVUCi*, represented by the *i*-th vulnerability criteria, can be calculated using the average value of correspondent VUEs, considering the risk factor of vulnerabilities as well. The formula to derive the *i*-th *ActVUC* is defined as:

$$ActVUC_i = RskVUC_i \times \frac{\sum_{j=1}^{n} ActVUE_j}{n}, \ \forall RskVUC_i \ \in \ [0, \ 10] \qquad (4)$$

where,

ActVUCij: the actual score of the *j*-th VUE associated with the i-th SRC
RskVUCi: the risk that corresponds to the *i*-th VUC
n: the number of VUE associated with the *i*-th VUC

Assurance target evaluation phase

The third phase of evaluation is responsible for the calculation of the overall assurance score for the assurance target. This is achieved by aggregating the score of the assurance criteria and perspectives at the following three levels of calculation.

Level 1. The first level is to obtain a summative assurance score for each assurance perspective by accumulating the correspondent assurance criteria.

For *SRP*, we define a metric *ActSRP* to present the overall security-requirement score of the assurance target. The formula is as follows:

$$ActSRP = \sum_{i=1}^{n} ActSRC_i \tag{5}$$

where,

$ActSRC_i$: the assurance score of the i-th SRC
n: the number of SRC

Correspondingly, the formula used for the calculation of the overall vulnerability score is presented below:

$$ActVUP = \sum_{i=1}^{n} ActVUC_i \tag{6}$$

where,

$ActVUC_i$: the assurance score of the i-th vulnerability criteria
n: the number of vulnerability criteria

Level 2. At the second level, the overall security assurance score (*SAS*) of the assurance target is derived from the difference between the security-requirement score (*ActSRP*) and vulnerability score (*ActVUP*). Thus, the formula for deriving the actual SAS (i.e., *ActSAS*) is presented as follows:

$$ActSAS = \ ActSRP - ActVUP \tag{7}$$

Level 3. It can be noticed that the scale of *SAS* is highly influenced by the number of security requirements as well as vulnerabilities included in the evaluation model (Equations 5 and 6). This leads to a variant range of assurance scores among different assurance targets, and further, makes it difficult to interpret to take decisions among various systems. In this regard, *SAS* must be normalized to a common scale for a more comprehensive and understandable value, named the security assurance level (*SAL*). We adopt the min-max normalization method (Jayalakshmi & Santhakumaran, 2011), which preserves the relationships among the original data values. This method will encounter an out-of-bounds error if a future input case for normalization falls outside the first data range for the attribute. The formula of this generic normalization method is presented as follows:

$$v' = \frac{v - \min_A}{\max_A - \min_A}(newmax_A - new\min_A) + newmin_A \tag{8}$$

where,

$\min_A$ and $\max_A$: the minimum and maximum values of an attribute

$newmin_A$ and $newmax_A$: the new minimum and maximum values after normalization
v: the old value of an attribute
v': the new value after normalization

The convention we follow for the *SAL* is that it lies in the interval between 0 and 10, where 0 corresponds to the worst possible level of security assurance, while 10 to the excellent assurance level. Thus, the formula for metric *SAL* can be defined as:

$$SAL = \frac{ActSAS - MinSAS}{MaxSAS - MinSAS} \times (10 - 0) + 0 = \frac{ActSAS - MinSAS}{MaxSAS - MinSAS} \times 10 \tag{9}$$

To derive *MaxSAS*, we can refer to Equation (7), from which we know that SAS can be maximum when the following two conditions are met:

1. All security requirements are fulfilled, which causes the value of *ActSRP* to be maximum, and,
2. All possible vulnerabilities do not exist. This makes *ActVUP* minimum (zero).

SAS, on the other hand, can become minimum (i.e., *MinSAS*) if (i) All protection mechanisms are ineffective to fulfill the defined security requirements (*ActSRP* is minimum), and (ii) all listed vulnerabilities are found to exist in the assurance target, and all have maximum risk value (*ActVUP* is maximum).

The outcome of the metrics aggregation, as stated previously, is the assignment of an assurance level that lies in the [0,10] interval. However, for better comprehensibility, a discrete rating is provided as well. Table 5 is adapted from the table of NVD Vulnerability Severity Ratings (W3C, n.d.). However, our table is showing the opposite, i.e., levels of security. This table can be used to convert the score to a textual representation.

Table 5. Assurance level.

Assurance level	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

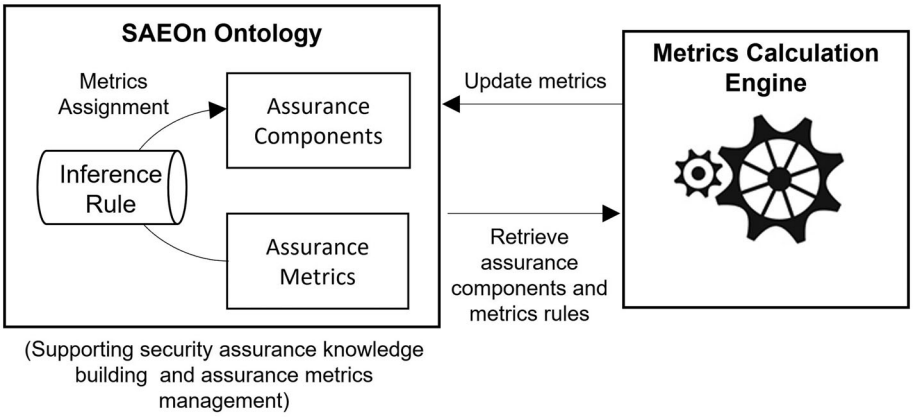


Figure 3. Overview of the ontology-based approach for generating security assurance metrics.

Ontology-based approach

Our ontology-based approach centers on the semantic modeling of security assurance components and metrics aggregation. It comprises a flexible architecture that includes a **S**ecurity **A**ssurance **E**valuation **On**tology (SAEOn) and a metrics calculation engine. The former is used to describe the SAE metamodel, while the latter is applied to combine ontology operations and metrics calculation. An overview of this approach is depicted in Figure 3. In a nutshell, SAEOn prescribes constructing a semantic model based on the methodology of the quantitative SAE that defines classes of assurance components (e.g., assurance targets, assurance criteria, etc.) and assurance metrics (e.g., security assurance levels, security requirement scores, etc.) that capture the security knowledge required for security assessment and analysis. Besides, all the assurance metrics and the corresponding calculation rules are expressed and stored by the ontology. The metrics are assigned to corresponding assurance components through inference rules and reasoning engines. The inference rules serve as a bridge for the connections between assurance components and assurance metrics. Our ontology-based approach aims to be entirely ontology-driven, where we employ the code-data separation principle for integrating SAEOn and metrics aggregation algorisms. The aim is that no domain/ontology-specific code is required on the engine side, and the processing and displaying metrics are steered by the ontology. The specific modules of the approach are described in the following sections.

SAEOn ontology

This study employs OWL (Web Ontology Language), a markup language based on W3C RDF/XML (W3C, 2012), as the notation for representing

SAEOn and adopts Protégé (Tudorache et al., 2013) as the OWL editing tool of the ontology construction. The OWL is a Semantic Web language designed to represent rich and complex knowledge about things, and relations between things, while Protégé is a free, open-source, and integrated development platform that provides a suite of facilities to create and manage ontological models. Using Protégé, the security assurance model presented in the previous section is easily transposed into an ontology that can be described in an equivalent XML-based format. Furthermore, Protégé can be extended by way of a plug-in architecture and a Java-based application programming interface (API) for building knowledge-based tools and applications (Zhao & Liu, 2008). In the following, we discuss the modeling approach regarding assurance components and assurance metrics, as well as the integration of the two models.

Assurance component modeling

In SAEOn, each component of the security assurance model is represented by one OWL *Class* having the same name and role. To have a finer classification and enable inference within the ontology, a *SubClassOf* type restriction is added to the model to represent the concepts of *Security Requirement* and *Vulnerability* accordingly. Figure 4 presents the class hierarchy in the Protégé.

Relationships between the classes are represented as *Object Property* in Protégé, following camel-case syntax naming conventions. Object properties

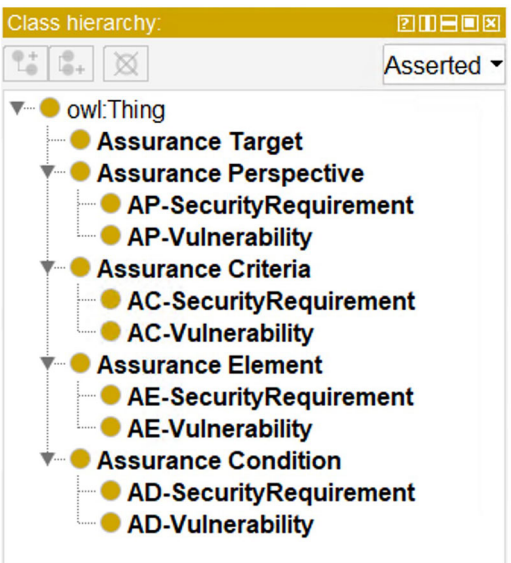


Figure 4. Classes of the security assurance ontology in Protégé.

Table 6. Objective properties of the security assurance ontology.

Object property	Domain	Range
hasAssurancePerspective	Assurance target	Assurance perspective
hasAssuranceCriteria	Assurance perspective	Assurance criteria
hasAssuranceElement	Assurance criteria	Assurance element
hasAssuranceCondition	Assurance element	Assurance condition

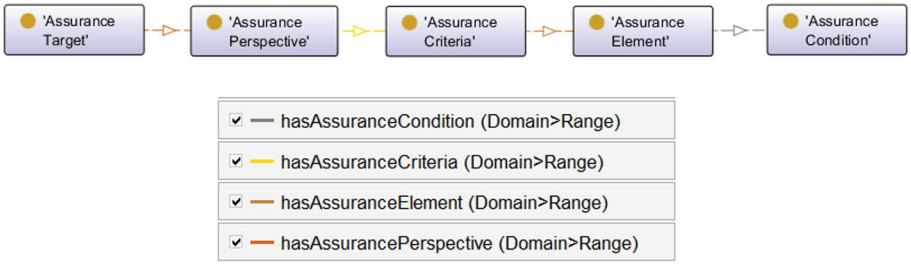


Figure 5. Illustration of relationships between classes.

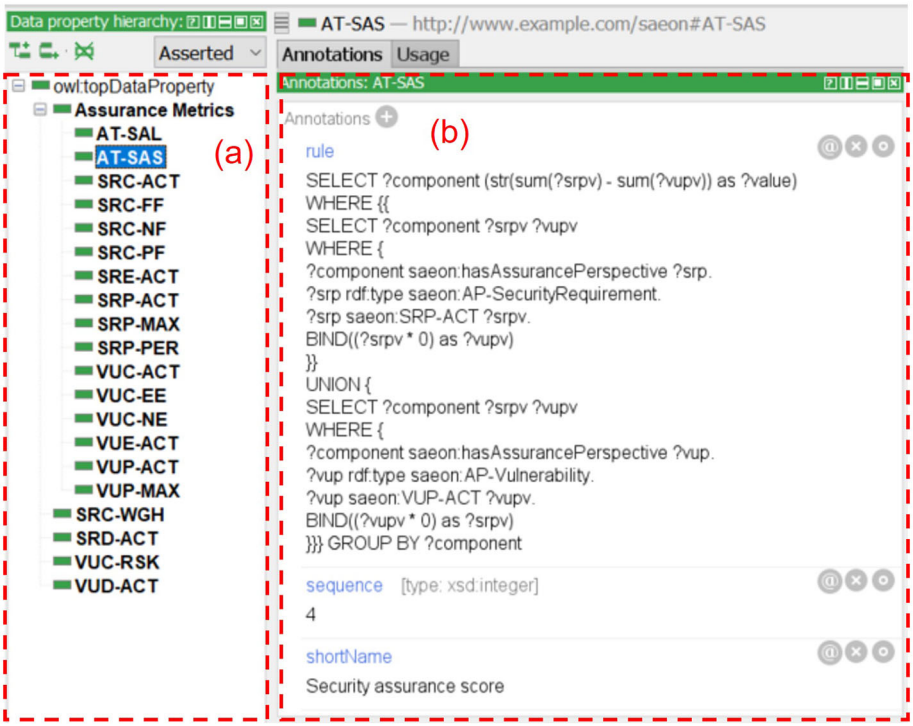


Figure 6. Configurations of data properties in Protégé.

refer to these properties with classes as both domains and ranges. Table 6 lists object properties in the ontology and their associated constraints, while Figure 5 illustrates the relationships between classes after the configuration, created using the OntoGraf plugin (Falconer, n.d.).

Assurance metrics modeling

The metric model includes a set of metrics, which is defined as a *Data Property* assertation in Protégé, with a numeric range. Figure 6 shows an example of the definition of a data property. The naming of data properties in SAEOn follows the pattern: s*egment1* "-" segment*2*, where *segment1* represents the assurance component, and *segment2* is the metric. For example, *AT-SAS* and *SRP-ACT*, represent "the security assurance score of the assurance target" and "the actual score of SRP," respectively. In OWL, data properties can be any data attribute with different data types (e.g., String, Literal, etc.). To distinguish those data properties needed calculation (i.e., assurance metrics) from others, we model the list of data properties hierarchically, in which an additional node *AssuranceMetrics* (also a data property) is created to categorize the metrics that needed calculation, depicted in Figure 6, area (a). In addition, with Protégé one can add annotations to any OWL entity, such as labels, descriptions, and cross-references (`xrefs`). With this feature, we create three customized annotation properties: *shortName*, *rule*, and s*equence* to represent the description, calculation rules, and calculation sequences of assurance metrics, respectively, also shown in Figure 6, area (b).

To make the code of the metrics calculation engine independent from SAEOn, the ontology must firstly model what calculation algorisms the metrics refer to. In this respect, a new annotation is created, named *rule*, which can be assigned to data properties flexibly. The content of the *rule* is a SPARQL statement used to retrieve and aggregate data from the ontology. SPARQL is a W3C-recommended semantic query language (W3C, 2013a), able to retrieve and manipulate data stored in RDF format (Segaran et al., 2009). The greatest strength of SPARQL is navigating relationships in RDF graph data through graph pattern matching. In addition, SPARQL supports aggregation expressions to select and return multiple result values after grouping query solutions in a certain way (e.g., COUNT, SUM, AVG, etc.). For the ontology-based approach, this is a powerful feature for exploiting repositories of assurance metrics, by allowing us not only to retrieve descriptive information (metadata) of the metrics but also formally describe the logic that can be executed directly by machines. With such a design, we can easily encode calculation algorisms for assurance metrics and centralized the maintenance in the ontology.

Through data properties, we have built a set of meaningful metrics upon those introduced in Section Assurance metrics computation. For example, the SPARQL for deriving the actual score of SRC (i.e., *ActSRC* in Equation 3) is shown in Listing 1. For abbreviating the URI (Uniform Resource Identifier) of the SAEOn ontology, we use `saeon` for Prefix declaration in SPARQL

Listing 1. The SPARQL for calculating the actual score of SRC (SRC-ACT).

```
SELECT ?component (str(AVG(?v) *AVG(?w)) as ?value)
WHERE {
      ?component rdf:type saeon:AC-SecurityRequirement.
      ?component saeon:hasAssuranceElement ?ae.
      ?ae saeon:SRE-ACT ?v.
      ?component saeon:SRC-WGH ?w.
}
GROUP BY ?component
```

Due to space limitations, only a few selected SPARQL rules are presented under this section, while others could have been excluded. Another example of metrics is *SRC-FF*, representing the "Numbers of fully fulfilled security requirements in SRC." The SPARQL to derive the metric is presented in Listing 2.

Listing 2. The SPARQL for calculating the numbers of full fulfilled security requirements in SRC (SRC-FF).

```
SELECT ?component (str(count(*)) AS ?value )
WHERE {
      ?at saeon:hasAssurancePerspective ?ap.
      ?ap saeon:hasAssuranceCriteria ?component.
      ?component sao:hasAssuranceElement ?ae.
      ?ae saeon:SRE-ACT ?v.
      ?component rdf:type saeon:AC-SecurityRequirement.
      FILTER(?v = 1)
}¨
GROUP BY ?component
```

To assess the overall performance of the security requirement perspective, we define a metric *SRP-PER*, calculated using the ratio between the actual score and maximum score of SRP, which are derived using the summation of *SRC-ACT* (Equation 5) and *SRC-WGH*, respectively. It is noted that the possible maximum score of SRC always equals its weight value. The SPARQL for calculating *SRP-PER* is presented in Listing 3.

Listing 3. The SPARQL for calculating the performance of SRP (SRP-PER).

```
SELECT ?component (str(sum(?v) / sum(?w)) AS ?value)
WHERE {
       ?component saeon:hasAssuranceCriteria ?ac.
       ?ac saeon:SRC-WGH ?w.
       ?ac saeon:SRC-ACT ?v.
       ?ap rdf:type saeon:AP-SecurityRequirement
}
GROUP BY ?component
```

The last example is *AT-SAS*, i.e., the security assurance score of an assurance target. The SPARQL is shown in Listing 4, in which a UNION statement is utilized to synthesize the scores of *AT-SAS*, *SRP-MAX*, and *VUP-MAX* from different result sets.

Listing 4. The SPARQL for calculating security assurance level (AT-SAS).

```
SELECT ?component
       (str((SUM(?sas)-SUM(?min)) / (SUM(?max) - SUM(?min)) * 10)AS ?value)
WHERE {{
    SELECT ?component ?sas ?min ?max
    WHERE {
       ?component saeon:AT-SAS ?sas.
       BIND((?sas * 0) AS ?min).
       BIND((?sas * 0) AS ?max).
       ?component rdf:type saeon:AssuranceTarget
    }}
UNION{
    SELECT ?component ?sas ?min ?max
    WHERE {
       ?component saeon:hasAssurancePerspective ?ap.
       ?ap rdf:type saeon:AP-SecurityRequirement.
       ?ap saeon:SRP-MAX ?max.
       BIND((?max * 0) AS ?sas).
       BIND((?max * 0) AS ?min).
    }}
UNION {
    SELECT ?component ?sas ?max ?min
    WHERE {
       ?component saeon:hasAssurancePerspective ?ap.
       ?ap rdf:type saeon:AP-Vulnerability.
       ?ap saeon:VUP-MAX ?vup.
       BIND((?vup * -1) AS ?min).
       BIND((?vup * 0) AS ?sas).
       BIND((?vup * 0) AS ?max).
}}}GROUP BY ?component
```

The next design consideration is the metrics calculation order, which should follow the three phases of the assurance evaluation (presented in Section Assurance metrics computation). In this respect, we use a customized annotation "*sequence*" to indicate the computing sequences of metrics along the metrics aggregation process. To reduce the dependence between metrics in the same evaluation phase, meanwhile, to simplify the metrics calculation sequencing, all the metrics are directly calculated based on those in the previous phase. With this design principle, the calculation sequence can be decided simply following the evaluation phase, except for *AT-SAS* and *AT-SAL*, which are set in the (extra) fourth and fifth phases because of their interdependence. Table 7 shows the completed list of the configured data properties, representing metrics names, descriptions, and calculation sequences. With the features described above, the metrics metadata

Table 7. Data properties and their annotations.

Data property	Short name	Sequence
AT-SAL	Security assurance level	5
AT-SAS	Security assurance score	4
SRP-PER	Performance of overall security requirements	3
SRP-MAX	Max. score of overall security requirements	3
SRP-ACT	Actual score of overall security requirements	3
SRC-WGH	Weight factor of SRC	
SRC-ACT	Actual score of SRC	2
SRC-FF	Number of full fulfilled security requirements in SRC	2
SRC-PF	Number of partially fulfilled security requirements in SRC	2
SRC-NF	Number of unfulfilled security requirements in SRC	2
SRE-ACT	Actual score of SRE	1
SRD-ACT	Actual score of SRD	
VUP-MAX	Max. score of overall security vulnerabilities	3
VUP-ACT	Actual score of overall security vulnerabilities	3
VUC-RSK	Risk factor of VUC	
VUC-ACT	Actual score of VUC	2
VUC-EE	Number of existing vulnerabilities in VUC	2
VUC-NE	Number of non-existed vulnerabilities in VUC	2
VUE-ACT	Actual score of VUE	1
VUD-ACT	Actual score of SRD	

```
<!-- http://www.example.com/saeon#AP-SecurityRequirement -->

<owl:Class rdf:about="http://www.example.com/saeon#AP-SecurityRequirement">
    <rdfs:subClassOf rdf:resource="http://www.example.com/saeon#AssurancePerspective"/>
    <saeon:shortName>Security Requirement</saeon:shortName>
</owl:Class>
```

Figure 7. Encoding classes in RDF/XML.

structured in the ontology is characterized as a repository of metrics with semantic descriptions and flexible metrics configuration.

Encoding assurance components and metrics

Assurance components and metrics are encoded in OWL such that they can be understandable to both machines and human beings. OWL provides RDF/XML syntax to represent ontology-based domain knowledge, which also provides a platform-independent, Internet-based interaction of domain experts with SAEOn. More importantly, as a formal language with description logic-based semantics, OWL enables automatic reasoning about inconsistencies of concepts. Figure 7 shows the part of the RDF/XML file that encodes an OWL class *AP-SecurityRequirement*. The construct `rdfs:subClassOf` indicates that "*AP-SecurityRequirement*" is a subclass of *AssurancePerspective*. Metrics information is also encoded in RDF/XML syntax. For example, the encoding of the metric *AT-SAS* is depicted in Figure 8. The `owl:DatatypeProperty` is used to indicate that the represented property is a data property. The `rdfs:subPropertyOf` is used to identify data properties that are sub-properties of an RDF resource *AssuranceMetrics*.

```
<!-- http://www.example.com/saeon#AT-SAS -->

<owl:DatatypeProperty rdf:about="http://www.example.com/saeon#AT-SAS">
    <rdfs:subPropertyOf rdf:resource="http://www.example.com/saeon#AssuranceMetrics"/>
    <saeon:rule>SELECT ?component (str(sum(?srpv) - sum(?vupv)) as ?value)
      WHERE {{
      SELECT ?component ?srpv ?vupv
      WHERE {
      ?component saeon:hasAssurancePerspective ?srp.
      ?srp rdf:type saeon:AP-SecurityRequirement.
      ?srp saeon:SRP-ACT ?srpv.
      BIND((?srpv * 0) as ?vupv)
      }}
      UNION {
      SELECT ?component ?srpv ?vupv
      WHERE {
      ?component saeon:hasAssurancePerspective ?vup.
      ?vup rdf:type saeon:AP-Vulnerability.
      ?vup saeon:VUP-ACT ?vupv.
      BIND((?vupv * 0) as ?srpv)
      }}} GROUP BY ?component
    </saeon:rule>
    <saeon:sequence rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</saeon:sequence>
    <saeon:shortName>Security assurance score</saeon:shortName>
</owl:DatatypeProperty>
```

Figure 8. Encoding data properties in RDF/XML.

Modeling metrics assignment

To assign values for metrics, the designed metrics should be coupled with relevant assurance components. To succeed in reaching this task efficiently, we model the metrics assignment by semantic rules. Such rules are semantic because they use the domain-specific semantics of a data item. For this, Semantic Web Rule Language (SWRL) (W3C, 2004), an expressive OWL-based rule language is used which efficiently derives implicit facts from explicitly given ones. SWRL allows users to write rules that can be expressed in terms of OWL concepts to provide more powerful deductive reasoning capabilities than OWL alone. It mainly deals with Horn-like rules that are of the form of an implication between an antecedent (body) and consequent (head), meaning that the conditions specified in the consequent must hold whenever the conditions specified in the antecedent are satisfied. In this syntax, a rule has the form

$$Antecedent \rightarrow Consequent$$

where both antecedent and consequent are conjunctions of atoms. Variables are indicated using the standard convention of prefixing them with a question mark (e.g.,? x).

Based on the classes and properties that have been modeled in SAEOn, the rules of metrics assignment are expressed in SWRL. For example, the rule to assign an initial score to *AT-SAS* can be exemplified in SWRL as follows.

$$AssuranceTarget(?at) \rightarrow AT - SAS(?at, 0)$$

Table 8. Excerpt of SWRL rules used within the ontology.

No.	Rule
R01	AssuranceTarget(?t) → AT-SAL(?t, 0)
R02	AssuranceTarget(?t) → AT-SAS(?t, 0)
R03	AP-SecurityRequirement(?sr) → SRP-ACT(?sr, 0)
R04	AP-SecurityRequirement(?sr) → SRP-MAX(?sr, 0)
R05	AP-SecurityRequirement(?sr) → SRP-PER(?sr, 0)
R06	AC-SecurityRequirement(?sr) → SRC-ACT(?sr, 0)
R07	AC-SecurityRequirement(?sr) → SRC-WGH(?sr, 0)
R08	AC-SecurityRequirement(?sr) → SRC-FF(?sr, 0)
R09	AC-SecurityRequirement(?sr) → SRC-PF(?sr, 0)
R10	AC-SecurityRequirement(?sr) → SRC-NF(?sr, 0)
R11	AE-SecurityRequirement(?sr) → SRE-ACT(?sr, 0)
R12	AD-SecurityRequirement(?sr) → SRD-ACT(?sr, 0)
R13	AP-Vulnerability(?vu) → VUP-ACT(?vu, 0)
R14	AP-Vulnerability(?vu) → VUP-MAX(?vu, 0)
R15	AC-Vulnerability(?vu) → VUC-ACT(?vu, 0)
R16	AC-Vulnerability(?vu) → VUC-EE(?vu, 0)
R17	AC-Vulnerability(?vu) → VUC-NE(?vu, 0)
R18	AC-Vulnerability(?vu) → VUC-RSK(?vu, 0)
R19	AE-Vulnerability(?vu) → VUE-ACT(?vu, 0)
R20	AD-Vulnerability(?vu) → VUD-ACT(?vu, 0)

This rule implies that the range value of the data property *AT-SAS* will be given "*0*" for all OWL individuals that are members of the OWL class *AssuranceTarget*. In SAEOn, we configured 20 SWRL rules for data property inference. Table 8 presents these rules used within the ontology.

Metrics calculation engine

In this section, we describe our metrics calculation engine, which is designed as a kind of service that takes the SAEOn OWL file as input, interprets it, calculates metrics value, and returns a result set with the value for all assurance components. This engine is implemented in Java technologies and integrated with SAEOn ontology through the Apache Jena API (The Apache Software Foundation, 2011). Apache Jena is a free and open source Java framework for building Semantic Web applications, which helps developers develop code that handles semantic web building blocks, such as RDF and OWL in line with published W3C recommendations (W3C, 2013b). Figure 9 depicts the implementation of the metrics calculation engine, in which the functionality is provided through the cooperation of five modules.

OWL loader

This module is responsible for retrieving and parsing ontology elements from an input OWL file, meanwhile, loading the individuals, object properties, and data properties from the ontology. The programming interface Jena APIs are used to work with the ontology and the conversion to classes and objects is done by Java.

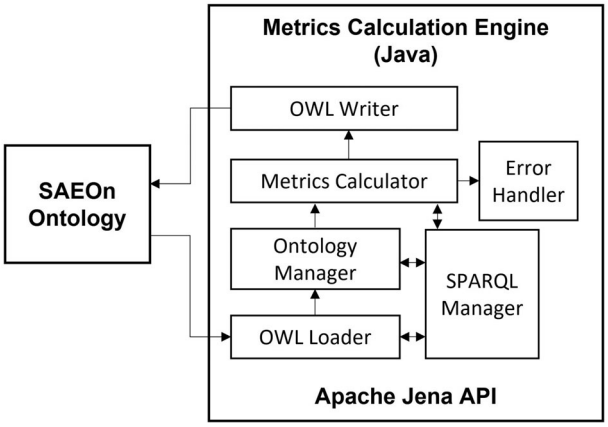


Figure 9. Implementation of the metrics calculation engine.

Ontology manager

this module manages access to the Java objects created from the *OWL Loader*. It sets a fusion of the assurance evaluation hierarchy to combine all ontology elements. Meanwhile, it provides linkages to Java classes providing the code executed when metrics are processing.

SPARQL manager

This module is the central component module that handles all SPARQL for actual querying the semantic model. Since the SPARQL of the metrics calculation rules are stored as strings in data properties, they can be executed directly by the SPARQL Manager without transformation. Besides, this module is responsible performs *Update* operations for RDF graphs in the ontology.

Metric calculator

This module executes SPARQL to derive aggregate metrics following the configuration. If errors occur during processing metrics, they will be registered by the *Error Handler*. For each successful metrics calculation, a SPARQL string is prepared for updating the specific RDF dataset. Listing 5 provides the code snippet for constructing the SPARQL.

Listing 5. Java code snippet for constructing the SPARQL.

```
StringSparql =
"DELETE { ?s ?p ?o  \n" +
"WHERE {?s ?p ?o. \n" +
"?p rdf:type owl:DatatypeProperty. \n" +
"FILTER regex(str(?s),"+ "\""+componentName+"\"" + "). \n" +
"FILTER regex(str(?p),"+ "\""+metricsName+"\"" + ")}; \n" +
"INSERT DATA { \n" +
ontologyName + ":" + objectName + " " + ontologyName + ":" + metricsName + " "
+ metricsValue + " }";
```

The update string consists of two operations, including a triple (i.e., the specific individual `componentName` and its data property `metricsName`) to be deleted and a new triple (used here to revise the metrics value `metricsValue`). The requested change happens in the name graph identified by the IRI, `ontologyName`, which is `saeon` in our case. The string will be concatenated recursively along the metrics calculation process and be taken into the *SPARQL Manager* at the end of the whole process for updating the semantic model, using the Jena API `UpdateAction.parseExecute(sparql, model)`.

Error handler

The module is responsible to handle errors thrown by the *Metrics Calculator*. All errors will be registered in a Java class `MetricsException`.

OWL writer

After the model update is successful, this module writes the semantic model back to the original OWL file with the API: `model.write(FileWriter(fileName),"RDF/XML)`.

Listing 6 provides a pseudo-code representation of the overall metrics calculation algorithm.

Listing 6. Pseudo-code representation of the assurance metrics calculation algorithm.

```
F ← given OWL file
AC ← null
model ← owlLoader(F)  // Read an OWL file
classIter ← model.listClasses()
while classIter.hasNext() do
   ind = classIter.getIndivisual()  // get individual
   if ind.calculatedMetrics = Y then // Check whether the metric should be processed
      N ← ind.getName()
      M ← ind.classIter.getMetrics()
      S ← ind.getSeq()
      R ← ind.getSparql()
      AC.Add(N, M, S, R)  // Add assurance components
   end if
end while
sort AC[] by S ascending  // Sort assurance components by the calculation sequence
for i ← 1 to AC.length do
   MV ← runSparql(AC[i].R)   //Run SPARQL and derive metrics value
   if MV != null then
      // Prepare SPARQL for updating OWL
      Q  ← prepareUpdateSparql((AC[i].N, AC[i].M, MV)
   end if
end for
model ← executeSparql(OntModel, Q)  // Execute SPARQL for updating OWL
writeOWL(model)   //Write OWL to the file
```

Evaluation

Preliminary evaluation of SAEOn

After the ontology is constructed, a preliminary evaluation was performed to test its feasibility. According to Gómez-Pérez (1994), ontology evaluation refers to the correct building of the content of ontology, ensuring that its definitions correctly implement the ontology requirements and competency questions. The goal of our ontology evaluation is placed on the competency of the ontology toward addressing the effectiveness and correction of quantitative SAE. Thus, the "fundamental" aspects of the developed ontology to be tested include (1) Task-based evaluation, and (2) Logical consistency evaluation.

Task-based evaluation

This evaluation aims to assess what domain tasks have to be supported by an ontology and how the ontology can be used to accomplish certain tasks (Raad & Cruz, 2015). To achieve this, it is necessary to fill the ontology with basic, "ground-level" items for a specific case in the SAE. In this evaluation, we took an exemplary assurance target in the domain of web applications, named *System1*, which is a cloud platform for creating and maintaining virtual servers and networks. For this to work, we leveraged OWASP as the knowledge source to model the knowledge items in the domain of web applications. Three OWASP project materials are chosen: OWASP ASVS, OWASP Top 10, and Web Security Testing Guide (WSTG) (OWASP, 2020). The first material is used to construct knowledge items for security requirements, while the last two are synthesized for the vulnerability items.

Individual creation and assurance metrics assignment. Once the domain knowledge is collected, we create OWL *individuals* for modeling the knowledge items. The individual generation is the process to create ontology instances based on classes of SAEOn, which constitutes a part of the ontology configuration. Moreover, through the *Property Assertions* function in Protégé, we can bind the classes with their respective individuals, so we could perform queries to extract the knowledge items in the ontology. For now, the creation for OWL individuals is done manually in Protégé, as well as the object properties (relationships) between individuals. Where modeling individuals for all OWASP items is rather time-consuming, in this study, we model a limited dataset instead of a full-scope implementation. Ongoing work is being carried out to simplify this maintenance. Figure 10

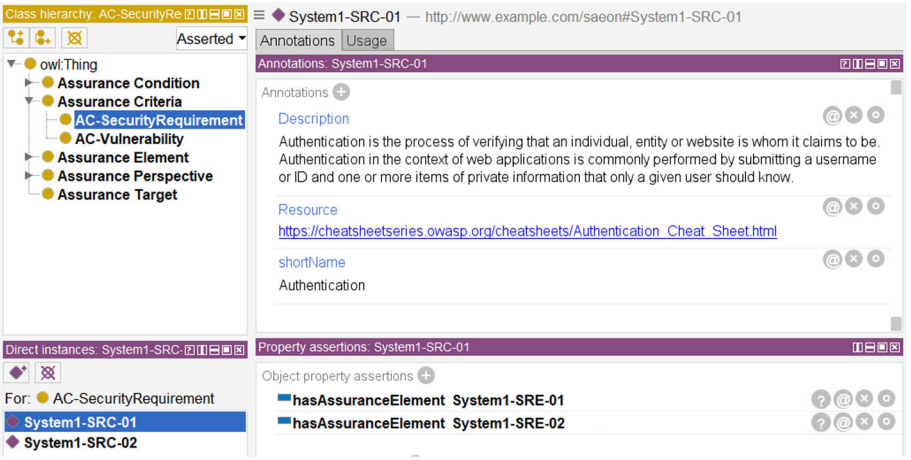


Figure 10. Configuration of individuals in Protégé.

demonstrates the configuration of an individual under the class of *AC-SecurityRequirement*, named *System1-SRC-01*. This individual (i.e., a specific security requirement criteria) is declared with two corresponding assurance elements. We use customized annotations to provide auxiliary information about the individual, for example, *shortName* and *Resource*. Furthermore, all assurance conditions are given an initial score (in data properties *SRD-ACT* and *VUD-ACT*) randomly for later metrics calculation.

To assign the metrics to the corresponding individuals, it is required to process the inference from SWRL rules with an appropriate rule engine. In Protégé, the Drools rule engine (O'Connor et al., 2005) is a plug-in to the SWRL that enables the processing of SWRL expression. The process of generating inferred knowledge using SWRL Drools Engine passes through 3 steps which are explained below:

1. OWL + SWRL → Drools: This step transfer SWRL rules and relevant OWL knowledge to the rule engine.
2. Run Drools: In this step, Drools runs the inference engine and generate new knowledge.
3. Drools → OWL: This step transfers the inferred rule engine knowledge to OWL knowledge.

Through executing the Drools engine, the metrics assigned to each assurance component are completed successfully. Figure 11 shows the inferred results of data properties in different individuals (*System1* and *System1-SRC-01*).

Competency-question evaluation. After the ontology is filled, we took a competency question (CQ) approach, through answers to SPARQL queries

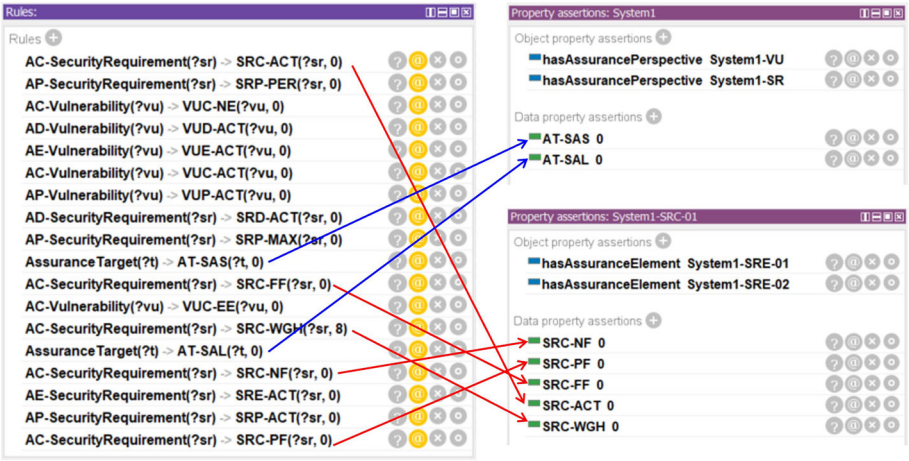


Figure 11. SWRL Rules in Protégé and the corresponding inferences.

executed as part of performing tasks. For an ontology to be considered useful, it must be able to give reliable answers to domain-specific questions using its terminology. The CQ-SPARQL method of evaluation is considered a very effective evaluation technique to test the adaptability and consistency of an ontology (Raad & Cruz, 2015). If the SPARQL queries can extract individuals as a response, it signifies that the CQs have succeeded in covering the defined objectives of the ontology. Therefore, three exemplary CQs were developed considering how the ontology fulfills the use cases.

CQ 1. List all assurance components relates to security requirements for the assurance target "*System1*" and mark the assigned score for assurance conditions.

This CQ aims to test the hierarchical structure of assurance components for a given assurance target, with linkage to the annotation and data properties. To answer this CQ, we extract individuals following the hierarchical relationships between them and filter the result using the given assurance target (*System1*) and the assurance perspective (*Security Requirement*). The corresponding SPARQL statement the CQ and the execution result in the Protégé are depicted in Figure 12.

CQ 2. List the metrics to be calculated with the corresponding SPARQL for assurance components of SRC and SRE, sorting by the calculation sequence.

The purpose of this CQ is to verify the capability of synthesizing different assurance components and incorporating the metrics that needed to be processed. To answer CQ2, we use the RDF triple:? `metrics rdfs:subPropertyOf saeon:AssuranceMetrics` to extract assurance metrics to be processed while the data combination of SRC and

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX saeon: <http://www.example.com/saeon#>
SELECT DISTINCT ?ap ?ac ?ae ?ad (str(?mv) AS ?score)
WHERE {
        ?at saeon:hasAssurancePerspective ?AP.      ?AP saeon:hasAssuranceCriteria ?AC.
        ?AC saeon:hasAssuranceElement ?AE.          ?AE saeon:hasAssuranceCondition ?AD.
        ?AP saeon:shortName ?ap.                    ?AC saeon:shortName ?ac.
        ?AE saeon:shortName ?ae.                    ?AD saeon:shortName ?ad.
        ?AD ?metrics ?mv.
        ?metrics rdf:type owl:DatatypeProperty.
        FILTER(regex(str(?at),"System1") &&
        regex(str(?ap), "Security Requirement"))
} ORDER BY ?ac ?ae
```

ap	ac	ae	ad	score
"Security Requirement"	"Authentication"	"Credential Storage"	"Verify that passwords are stored in	"0.5"
"Security Requirement"	"Authentication"	"Credential Storage"	"Verify that the salt is at least 32 bits	"1"
"Security Requirement"	"Authentication"	"Password Security"	"Verify that user set passwords are ε	"0"
"Security Requirement"	"Authentication"	"Password Security"	"Verify that any printable Unicode ch	"0.8"
"Security Requirement"	"Authentication"	"Password Security"	"Verify that passwords of at least 64	"1"
"Security Requirement"	"Session Management"	"Session Binding"	"Verify the application only stores se	"1"
"Security Requirement"	"Session Management"	"Session Binding"	"Verify that session tokens possess	"0"
"Security Requirement"	"Session Management"	"Session Binding"	"Verify the application generates a n	"0"
"Security Requirement"	"Session Management"	"Session Termination"	"Verify that logout and expiration inv	"0"
"Security Requirement"	"Session Management"	"Session Termination"	"If authenticators permit users to rem	"1"
"Security Requirement"	"Session Management"	"Token-based Session M	"Verify the application allows users t	"1"

Figure 12. The SPARQL statement and the execution result of CQ 1.

SRE is achieved using a `UNION` syntax. Such SPARQL pattern will be extensively used by the metrics calculation engine to prepare the dataset before processing. The SPARQL statement and the result are shown in Figure 13.

CQ 3. List the assurance scores to provide an overview of the assurance evaluation result for the assurance target "*System1*."

After metrics are processed completely, the resulting data should be presented to a user. CQ3 is designed to verify the effectiveness of obtaining summarized scores at the level of assurance targets. To answer CQ 3, we first supply mockup data (i.e., metrics values) in data properties of assurance metrics. The SPARQL used to extract data properties and their corresponding range value for both assurance targets and assurance perspectives, as well as the SPARQL execution result, are shown in Figure 14.

Logical consistency evaluation

Finally, we evaluated SAEOn for consistency and conciseness to confirm that no contradictory facts existed based on the Description Logic (DL) (Gómez-Pérez, 1994). In this respect, we employed an in-built OWL DL reasoner, Pellet, to perform subsumption tests to check concept satisfiability and consistency. This reasoner tests for implied subclass relationships based on user-defined class relationships. Errors in the ontology were pointed out via error messages and inconsistent classes were marked "red" for review.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX saeon: <http://www.example.com/saeon#>
SELECT ?component ?metrics ?sequence ?SPARQL
WHERE {{
        SELECT DISTINCT ?component ?metrics (str(?seq) AS ?sequence) ?SPARQL
        WHERE {
                ?component rdf:type saeon:AC-SecurityRequirement.
                ?component ?metrics ?v.
                ?metrics rdf:type owl:DatatypeProperty.
                ?metrics rdfs:subPropertyOf saeon:AssuranceMetrics.
                ?metrics saeon:rule ?SPARQL.
                ?metrics saeon:sequence ?seq       }
} UNION {
        SELECT DISTINCT ?component ?metrics (str(?seq) AS ?sequence) ?SPARQL
        WHERE {
                ?component rdf:type saeon:AE-SecurityRequirement.
                ?component ?metrics ?v.
                ?metrics rdf:type owl:DatatypeProperty.
                ?metrics rdfs:subPropertyOf saeon:AssuranceMetrics.
                ?metrics saeon:rule ?SPARQL.
                ?metrics saeon:sequence ?seq       }
}} ORDER BY ?sequence
```

component	metrics	sequence	SPARQL
System1-SRE-05	SRE-ACT	"1"	"SELECT ?component (str(AVG(?v)) AS ?value) WHERE { ?component saeon:hasAssurance
System1-SRE-04	SRE-ACT	"1"	"SELECT ?component (str(AVG(?v)) AS ?value) WHERE { ?component saeon:hasAssurance
System1-SRE-03	SRE-ACT	"1"	"SELECT ?component (str(AVG(?v)) AS ?value) WHERE { ?component saeon:hasAssurance
System1-SRE-02	SRE-ACT	"1"	"SELECT ?component (str(AVG(?v)) AS ?value) WHERE { ?component saeon:hasAssurance
System1-SRE-01	SRE-ACT	"1"	"SELECT ?component (str(AVG(?v)) AS ?value) WHERE { ?component saeon:hasAssurance
System1-SRC-02	SRC-PF	"2"	"SELECT ?component (str(count(*)) AS ?value) WHERE { ?at saeon:hasAssurancePerspecti
System1-SRC-01	SRC-PF	"2"	"SELECT ?component (str(count(*)) AS ?value) WHERE { ?at saeon:hasAssurancePerspecti
System1-SRC-02	SRC-NF	"2"	"SELECT ?component (str(count(*)) AS ?value) WHERE { ?at saeon:hasAssurancePerspecti
System1-SRC-01	SRC-NF	"2"	"SELECT ?component (str(count(*)) AS ?value) WHERE { ?at saeon:hasAssurancePerspecti
System1-SRC-02	SRC-FF	"2"	"SELECT ?component (str(count(*)) AS ?value) WHERE { ?at saeon:hasAssurancePerspecti
System1-SRC-01	SRC-FF	"2"	"SELECT ?component (str(count(*)) AS ?value) WHERE { ?at saeon:hasAssurancePerspecti
System1-SRC-02	SRC-ACT	"2"	"SELECT ?component (str(AVG(?v)) AS ?value) WHERE { ?at saeon:hasAssurancePerspect
System1-SRC-01	SRC-ACT	"2"	"SELECT ?component (str(AVG(?v)) AS ?value) WHERE { ?at saeon:hasAssurancePerspect

Figure 13. The SPARQL statement and the execution result of CQ 2.

Since the final version of the ontology was devoid of DL errors, it can be concluded that the SAEOn ontology is consistent and satisfactory. We also used Pellet to check the conciseness of the ontology by running tests to compute the inferred object properties. Our tests did not show any redundant arc in the Protégé, hence the SAEOn ontology was deemed to be concise.

Evaluation of the ontology-based approach

To demonstrate and evaluate the proposed ontology-based approach, we have developed a prototyped web application for syntactic and semantic transformations of the security assurance data. Following the design presented in the previous sections, this application supports the calculation of the metrics defined in SAEOn as well as the presentation of the results. The general system architecture is depicted in Figure 15. The metrics calculation engine, involving Java technologies, including Servlets, JSP, AJAX, and Jena APIs, was deployed on an Apache Tomcat web server (version 9.0.30). The SAEOn OWL file is uploaded into the application through a

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX saeon: <http://www.example.com/saeon#>

SELECT DISTINCT ?at ?metrics ?shortName ?value
WHERE { {
        SELECT DISTINCT ?at ?metrics ?shortName (str(?mv) AS ?value)
        WHERE {
                ?at saeon:hasAssurancePerspective ?AP.
                ?AP saeon:shortName ?ap.
                ?AP ?metrics ?mv.
                ?metrics rdf:type owl:DatatypeProperty.
                ?metrics saeon:shortName ?shortName  }
}UNION{
        SELECT DISTINCT ?at  ?metrics ?shortName (str(?mv) AS ?value)
        WHERE {
                ?at saeon:hasAssurancePerspective ?AP.
                ?at ?metrics ?mv.
                ?metrics rdf:type owl:DatatypeProperty.
                ?metrics saeon:shortName ?shortName  }
} FILTER(regex(str(?at),"System1")).
} ORDER BY ?ap ?ac
```

at	metrics	shortName	value
System1	VUP-ACT	"Actual score of overall vulnerabilities"	"5.38"
System1	SRP-MAX	"Max. possible score"	"14.0"
System1	SRP-PER	"Performance of overall securiry requirements"	"0.652"
System1	SRP-ACT	"Actual score of overal security requirements"	"9.134"
System1	AT-SAS	"Security assurance score"	"3.754"
System1	AT-SAL	"Security assurance level"	"5.121"

Figure 14. The SPARQL statement and the execution result of CQ 3.

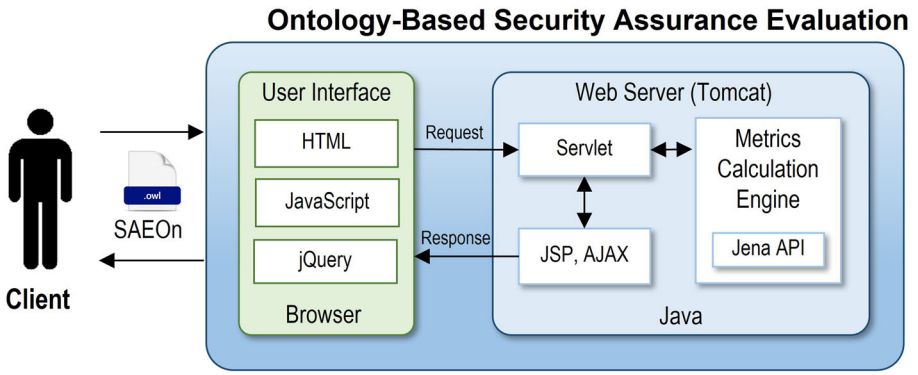


Figure 15. Implementation architecture of the prototyped ontology-based application.

browser-based user interface, which is developed using HTML, JavaScript, and jQuery libraries; through it, users can access the ontology data as well.

The user interface of the prototyped application is presented in Figure 16, which consists of four areas: OWL File Input, Assurance Metrics, Assurance Component, and Assurance Target Information. Initially, users choose a SAEOn OWL file, followed by two options: *Query* or *Run metrics*

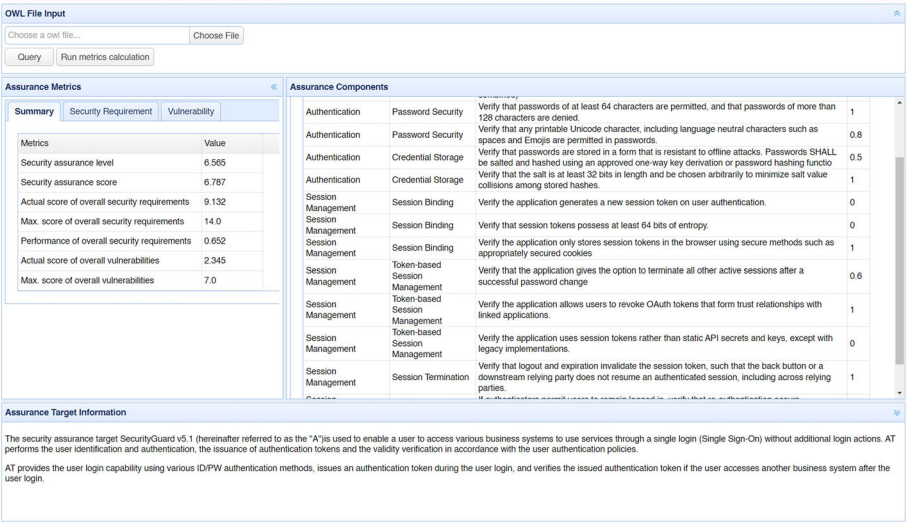


Figure 16. The Prototyped application using the ontology-based approach.

calculation. The former function retrieves and displays as-is ontology data (before calculation), whereas the latter triggers the metrics calculation engine to process metrics, update the owl file, and publish the result on the webpage.

To provide a categorical analysis of assurance metrics, we designed three different tabs for data presentation: the *Summary* tab, the *Security Requirement* tab, and the *Vulnerability* tab. The *Summary* tab shows the overall assurance metrics at the level of an assurance target. For example, the security assurance score of *System1* is 6.787, while the normalized assurance level is calculated as 6.565. In the *Security Requirement* and *Vulnerability* tabs, the metrics are presented with two separate tables, which are *Criteria Analysis* and *Element Analysis*, depicted in Figures 17 and 18, respectively. Each table includes components and corresponding metrics that are generated using the SPARQL of CQ3 in the previous section.

In this prototyped application, we also include the assurance component information and the initial score of assurance conditions before data aggregation (in the area of the *Assurance Component*), in which the dataset is retrieved using the SPARQL in CQ1. Moreover, the bottom area displays the description of the assurance target, which provides users with relevant and helpful background information on the assurance evaluation. This data is retrieved from the data property *Description* configured in the individual *System1*.

To exemplify the feature of the dynamic metrics assignment in our approach, we took another case-study evaluation, involving two scenarios: (1) adding a new metric, and (2) changing the metric rule. For the first scenario, it is assumed that a new metric is required for identifying the

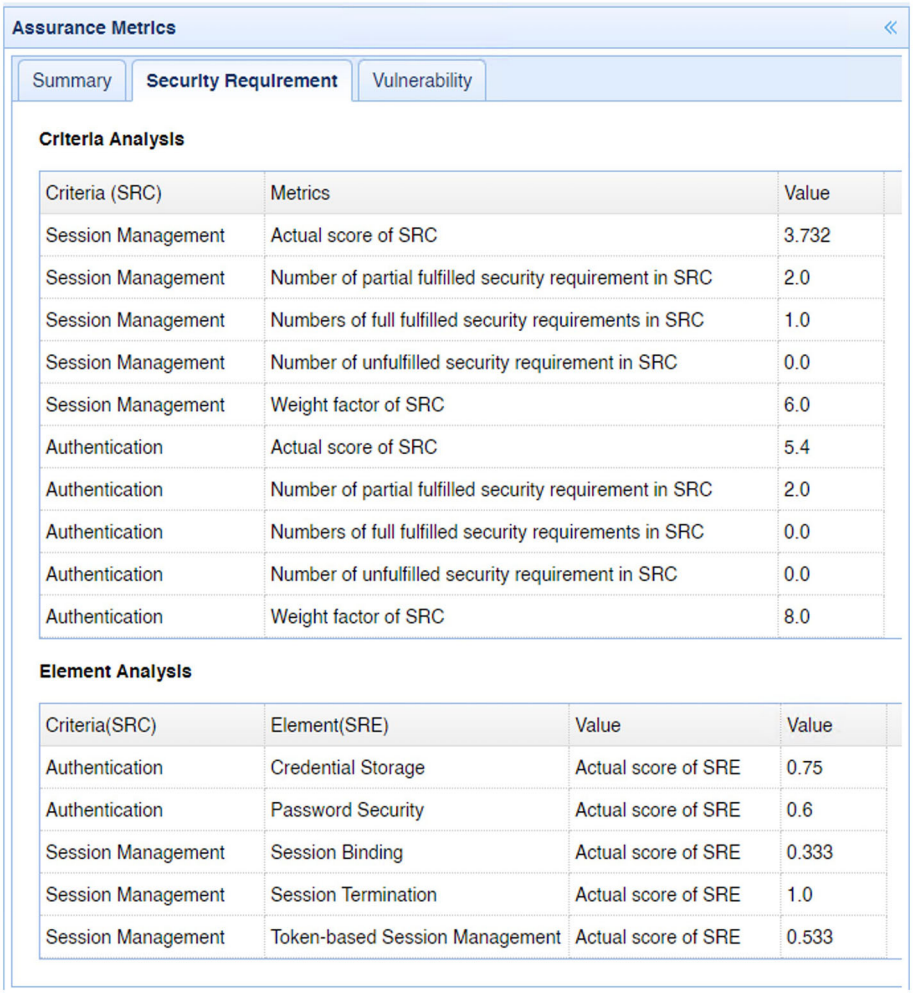


Figure 17. The user interface for security requirement metrics.

weak SRC, that is, not performing well in security requirement fulfillment. In this respect, we modeled a new data property in Protégé, named *SRC-PRS*, which represents the metrics "*Priority score of SRC*." In the second scenario, we changed the calculation rule for the actual score of vulnerability criteria (*VUC-ACT*), from the *Average* of SRE to the *Summation* of SRE. Subsequently, we modify the SPARQL statement in the annotation rule for the data property *VUC-ACT*. The configurations for the two metrics are depicted in Figures 19(a,b) separately. After applying SWRL rules and the Droops reasoning engine in Protégé, the new data property (*SRC-PRS*) was plugged into all SRC automatically. Thereafter, we uploaded the revised OWL file into the application, and run the metrics calculation. The calculation result is shown in Figure 20, in which (a) indicates the new metrics assignment for *SRC-PRS*, while (b) shows that *VUC-ACT* is derived using the new rule. The case study is extremely simple, but this is a good

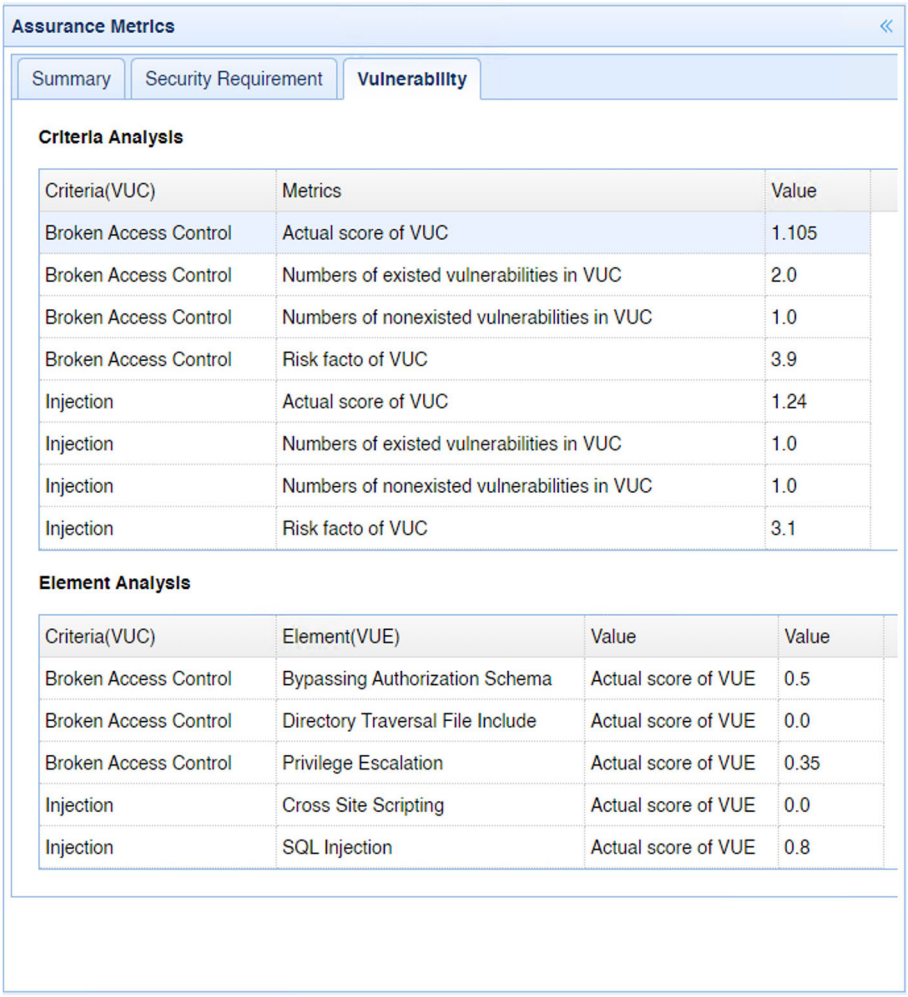
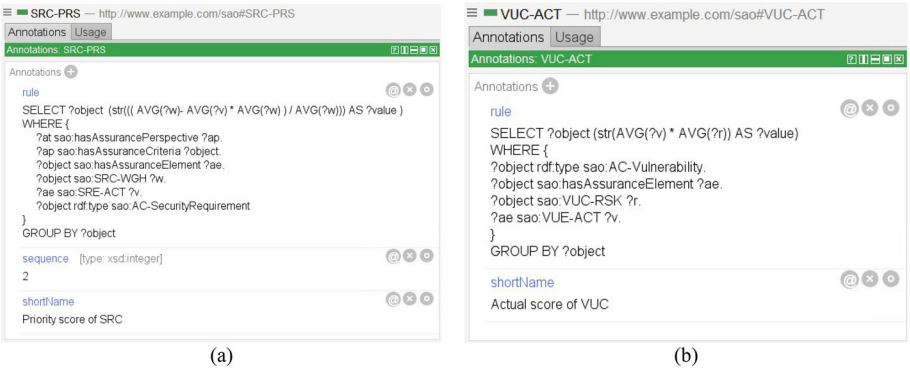


Figure 18. The user interface for vulnerability metrics.



(a) (b)

Figure 19. The data property configurations for the case study.

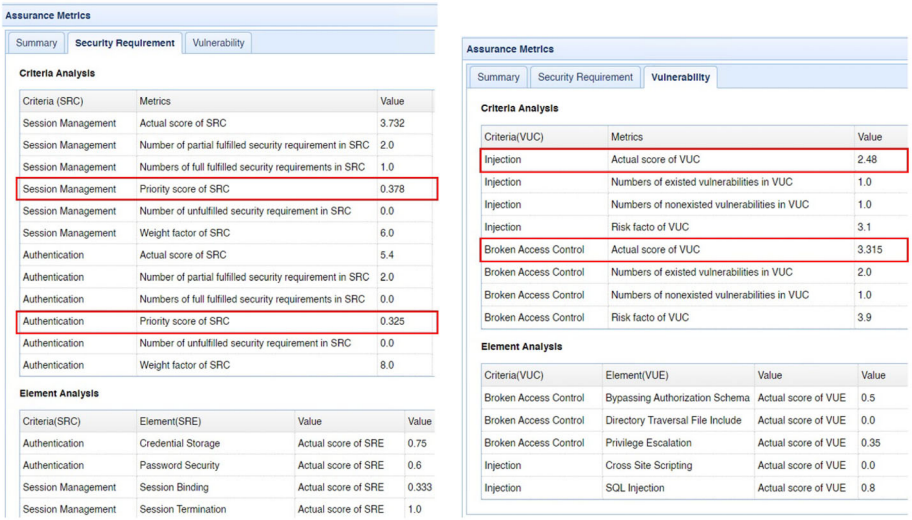


Figure 20. The result of the case study (scenario).

example to show how flexible our ontology-based approach is in managing assurance metrics.

Conclusion and future work

In this paper, we have presented an ontology-based approach for metrics computation in the system security assurance evaluation, which consists of (1) a quantitative SAE approach, (2) an ontology (SAEOn) for modeling the assurance components and metrics, and (3) a metrics calculation engine for automatically generating metrics values. The core principle behind the ontology-based approach is that SAE should be quantitative in nature and include both perspectives, the positive scores of the security, i.e., security requirement fulfillment, and the negative scores of the security, i.e., threat and vulnerability existence. The feasibility and effectiveness of the proposed approach were examined through a preliminary SAEOn evaluation and an application-based evaluation using the system prototype. The initial implementation and evaluation show that the quantitative SAE approach leveraging semantic web technologies is practical, which proves the effectiveness of the proposed method. Although the evaluation was only conducted on one application domain, it is obvious that the proposed ontology-based approach is highly customizable that can be applied in more application domains. Moreover, it is not restricted to a specific facet of security assurance, nor a particular security standard or framework.

Regarding the aim of our research, we argued that a well-designed repository of security assurance components, assurance metrics, and a

powerful knowledge management system can be effectively used to support SAE, such as for security-requirement elicitation, vuln discovery, metrics selection, and assurance score calculation and analysis, amongst others. This ontology allows us to define the entities of the security assurance model and the assurance metrics separately and relate them to each other in a modular way. With the formal definition of assurance components, the constructed knowledge can be reused, shared, and exchanged over time. Therefore, having an explicit security assurance ontology embedding these features was a key requirement for our proposal. Moreover, the proposed ontology has proven to be the foundation in the designing and prototypical implementation of the SAE application with semantic web power. The application prototype integrates a quantitative SAE framework with the semantic web via the proposed SAEOn ontology. The ontology created for the system not only provides re-usable content for the future SAE with similar purposes but also represents the hierarchical structure as a metamodel with more expressive relations. The standards provided by OWL have made it trivial to understand the semantics of the ontology.

SAE evaluators can use our proposed ontology to build the knowledge base of SAE cases, which could contain historical records, for example, system versioning, previous evaluation results, case comparison, and assurance components suitable for reuse. An advantage of using this type of ontological knowledge base approach is that it makes it easier to work with various types of assurance targets, which could correspond to actual contexts, among multiple cases. Such systematic methods in managing SAE cases using the ontology help the assurance evaluator to work with various types of assurance targets, meanwhile, construct SAE profiles efficiently. Moreover, this paper has provided an end-to-end solution to implement ontology-driven applications in quantitative SAE. Technical architecture is discussed in this paper that supports our proposed SAE methodology, following the predefined ontological metamodel. This research may inform researchers of the potential value of ontologies in hierarchical assurance metrics management, particularly for dynamic metrics configuration and aggregation. In addition, the proposed solution may be used for other similar purposes where ontologies are used to generate and present content in a metrics-driven analysis platform.

In summary, our ontology-based approach has the following advantages.

Generic and simple

To ease information overload while dealing with the huge amount of ontological knowledge content, the SAEOn ontology is created in an ingenious manner (five OWL classes), while it is still capable to model assurance

knowledge items and custom metrics in any application domain. In addition, the metrics calculation engine is designed to simply invoke the ontology-supplied aggregation rules inside the calculator. This implied that to measure new assurance components it is not necessary to add any code to the calculation engine.

Flexibility and extensibility

Ontologies have advantages for the reconstruction and expansion of knowledge, and it is easy to add new knowledge. In our approach, all assurance metrics are defined on the metamodel elements (data properties and annotations), and the rule can be represented using both semantic queries (*subject-predicate-object*) and aggregate functions, such as COUNT, SUM, and MIN. In particular, the rulesets for computing the metrics are not fixed, they can be completely customizable and highly adaptable to stakeholders' needs. Furthermore, the metadata is encoded in an XML format that has features of semantic richness and simplicity. As all metrics-related metadata are centralized in an ontology, they can be extended or modified in a rather flexible way.

Our prototyped application currently provides an eager implementation for demonstrating the assurance metrics calculation capability, which is independent of the assurance component maintenance. As we mentioned in Section Preliminary evaluation of SAEOn, facing a huge amount of knowledge items in the cyber security domain, is a complicated and time-consuming task for modeling individual as well as their relationships in the Protégé tool. It would, on the one hand, be possible to integrate the maintenance of knowledge items into the web application with well-designed user interfaces. On the other hand, a concept of "Assurance Profile" (Katt & Prasher, 2019) should be introduced to prepare a set of security knowledge for application domains of related products or services, which can be reused among SAE cases. Another area of future work is the definition of more practical security assurance metrics integrated into a Security Assurance Analytics (SAA) application, as proposed by Wen et al. (2022). Such an application supports the discovery, interpretation, and communication of meaningful patterns in data that allows management to measure the achievement of current activities of the organization in comparison to planned goals or outcome (Kaplan & Norton, 2005). The development of SAA for SAE brings several advantages, such as increased confidence in the level of assurance of the evaluation results, and possible automation of the evaluation process. Such an assurance evaluation technique will also contribute to promoting greater trust and transparency in the system's security. Furthermore. it is acknowledged that comprehensive testing of the

approach and extensive application to the wider cybersecurity domain will be required to further help validate both the ontology and the whole proposal.

Funding

This research work is financially supported by the Research Council of Norway through the SFI-Norwegian Centre for Cybersecurity in Critical Sectors (NORCICS, NFR project number: 310105).

ORCID

Shao-Fang Wen http://orcid.org/0000-0002-6228-8367

References

Agrawal, A., Alenezi, M., Khan, S. A., Kumar, R., & Khan, R. A. (2019). Multi-level fuzzy system for usable-security assessment. *Journal of King Saud University-Computer and Information Sciences*, *34*(3), 657–665.

Aman, W., & Khan, F. (2019). Ontology-based dynamic and context-aware security assessment automation for critical applications. In *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)* (pp. 644–647). IEEE. https://doi.org/10.1109/GCCE46687.2019.9015599

Anderson, R. (2020). *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons.

Arindaeng, K., Laboriante, A., Lu, Z. J., & Ragavendran, V. (2018). Indoor UAV tracking system.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, *284*(5), 34–43. https://doi.org/10.1038/scientificamerican0501-34

Białas, A. (2013). Ontology based model of the common criteria evaluation evidences. *Theoretical and Applied Informatics*, *25*(2), 69–91.

Bosch, J., Chiang, H.-F., & Gower, M. (2012). *LDM-503-2 (HSC reprocessing) test report*. Retrieved July 31, 2022, from https://dmtr-51.lsst.io/DMTR-51.pdf

Burns, S. F. (2005). Threat modeling: A process to ensure application security. *GIAC security essentials certification (GSEC) practical assignment*.

Doynikova, E., Fedorchenko, A., & Kotenko, I. (2020). A semantic model for security evaluation of information systems. *Journal of Cyber Security and Mobility*, *9*(2), 301-329. https://doi.org/10.13052/jcsm2245-1439.925

Ekclhart, A., Fenz, S., Goluch, G., & Weippl, E. (2007). Ontological mapping of common criteria's security assurance requirements. In *IFIP International Information Security Conference* (pp. 85–95). Springer.

Falconer, S. (n.d.). *Protege OntoGraf*. Retrieved February 2, 2022, from https://protegewiki.stanford.edu/wiki/OntoGraf

Fenz, S., & Ekelhart, A. (2009). Formalizing information security knowledge. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security* (pp. 183–194). ACM. https://doi.org/10.1145/1533057.1533084

Fernandez, E. B., Yoshioka, N., Washizaki, H., & VanHilst, M. (2010). Measuring the level of security introduced by security patterns. In *2010 International Conference on Availability, Reliability and Security* (pp. 565–568). IEEE.

Forum of Incident Response and Security Teams (2012). *CVSS*. Retrieved January 30, 2022, from https://www.first.org/cvss/

Franco Rosa, F., Jino, M., & Bonacin, R. (2018). Towards an ontology of security assessment: A core model proposal. In *Information technology–New generations*. Springer.

Gao, J., Zhang, B., Chen, X., & Luo, Z. (2013). Ontology-based model of network and computer attacks for security assessment. *Journal of Shanghai Jiaotong University*, *18*(5), 554–562. https://doi.org/10.1007/s12204-013-1439-5

Gómez-Pérez, A. (1994). From knowledge based systems to knowledge sharing technology: Evaluation and assessment.

Gonzalez-Gil, P., Skarmeta, A. F., & Martinez, J. A. (2019). Towards an ontology for IoT context-based security evaluation. In *2019 Global IoT Summit (GIoTS)* (pp. 1–6). IEEE. https://doi.org/10.1109/GIOTS.2019.8766400

Gritzalis, D., Karyda, M., & Gymnopoulos, L. (2002). Elaborating quantitative approaches for IT security evaluation. *Security in the Information Society*, *86*, 67–77.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, *5*(2), 199–220. https://doi.org/10.1006/knac.1993.1008

Herrmann, D. S. (2002). *Using the common criteria for IT security evaluation*. Auerbach Publications.

Heyman, T., Scandariato, R., Huygens, C., & Joosen, W. (2008). Using security patterns to combine security metrics. In *2008 Third International Conference on Availability, Reliability and Security* (pp. 1156–1163). IEEE. https://doi.org/10.1109/ARES.2008.54

Jayalakshmi, T., & Santhakumaran, A. (2011). Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, *3*, 1793–8201.

Kaplan, R. S., & Norton, D. P. (2005). The balanced scorecard: measures that drive performance. *Harvard Business Review*, *83*, 172.

Katt, B., & Prasher, N. (2018). Quantitative security assurance metrics: REST API case studies. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings* (pp. 1–7).

Katt, B., & Prasher, N. (2019). Quantitative security assurance. In *Exploring security in software architecture and design*. IGI Global.

Koinig, U., Tjoa, S., & Ryoo, J. (2015). Contrology-an ontology-based cloud assurance approach. In *2015 IEEE 24th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises* (pp. 105–107). IEEE.

Kotenko, I., Polubelova, O., Saenko, I., & Doynikova, E. (2013). The ontology of metrics for security evaluation and decision support in SIEM systems. In *2013 International Conference on Availability, Reliability and Security* (pp. 638–645). IEEE.

Liu, Y., & Jin, Z. (2015). SAEW: A security assessment and enhancement system of wireless local area networks (WLANs). *Wireless Personal Communications*, *82*(1), 1–19. https://doi.org/10.1007/s11277-014-2188-y

Maroc, S., & Zhang, J. B. (2019). Context-aware security evaluation ontology for cloud services. In *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (pp. 1012–1018). IEEE. https://doi.org/10.1109/IAEAC47372.2019.8997783

McGraw, G., Chess, B., & Migues, S. (2009). *Building security in maturity model*. Fortify & Cigital.

Mirante, D., & Cappos, J. (2013). Understanding password database compromises (Tech. Rep. TR-CSE-2013-02). Department of Computer Science and Engineering Polytechnic Institute of NYU.

O'Connor, M., Knublauch, H., Tu, S., Grosof, B., Dean, M., Grosso, W., & Musen, M. (2005). Supporting rule system interoperability on the semantic web with SWRL. In *International Semantic Web Conference* (pp. 974–986). Springer.

Ouedraogo, M. (2012). Towards security assurance metrics for service systems security. In *International Conference on Exploring Services Science* (pp. 361–370). Springer.

Ouedraogo, M., Khadraoui, D., Mouratidis, H., & Dubois, E. (2012). Appraisal and reporting of security assurance at operational systems level. *Journal of Systems and Software*, *85*(1), 193–208. https://doi.org/10.1016/j.jss.2011.08.013

Ouedraogo, M., Mouratidis, H., Khadraoui, D., & Dubois, E. (2009). Security assurance metrics and aggregation techniques for it systems. In *2009 Fourth International Conference on Internet Monitoring and Protection* (pp. 98–102). IEEE. https://doi.org/10.1109/ICIMP.2009.24

Ouedraogo, M., Reijo, M., Savola, H., Mouratidis, D., Preston, D., Khadraoui, D.., & Dubois, E.. (2013). Taxonomy of quality metrics for assessing assurance of security correctness. *Software Quality Journal*, *21*(1), 67–97. https://doi.org/10.1007/s11219-011-9169-0

OWASP. (2017). *Software assurance maturity model (SAMM)*. Retrieved June 3, 2022, from https://www.opensamm.org/2017/04/owasp-samm-v1-5-released/

OWASP. (2020). *Web security testing guide (WSTG)*. Retrieved June 3, 2022, from https://owasp.org/www-project-web-security-testing-guide/

OWASP (2021a). *Application security verification standard (ASVS)*. Retrieved June 3, 2022, from https://owasp.org/www-project-application-security-verification-standard/

OWASP (2021b). *OWASP top 10 application security risks*. Retrieved June 1, 2022, from https://owasp.org/www-project-top-ten/

Pham, N., & Riguidel, M. (2007). Security assurance aggregation for it infrastructures. In *2007 Second International Conference on Systems and Networks Communications (ICSNC 2007)* (pp. 72–72). IEEE. https://doi.org/10.1109/ICSNC.2007.75

Powley, S., Perry, S., Holt, J., & Bryans, J. (2019). An evaluation ontology applied to connected vehicle security assurance. *INCOSE International Symposium*, *29*(1), 37–52. https://doi.org/10.1002/j.2334-5837.2019.00588.x

Raad, J., & Cruz, C. (2015). A survey on ontology evaluation methods. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development, Part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*.

Raskin, V., Hempelmann, C. F., Triezenberg, K. E., & Nirenburg, S. (2001). Ontology in information security: a useful theoretical foundation and methodological tool. In *Proceedings of the 2001 Workshop on New Security Paradigms* (pp. 53–59).

Reddy, N. (n.d.). *An excellent compilation of software testing concepts (manual testing)*.

Rodes, B. D., Knight, J. C., & Wasson, K. S. (2014). A security metric based on security arguments. In *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics* (pp. 66–72).

Ross, R. S. (2011). *Managing information security risk: Organization, mission, and information system view*.

Segaran, T., Evans, C., & Taylor, J. (2009). *Programming the semantic web: Build flexible applications with graph data*. O'Reilly Media, Inc.

Shaaban, A. M., Schmittner, C., Gruber, T., Mohamed, A. B., Quirchmayr, G., & Schikuta, E. (2019). Ontology-based model for automotive security verification and validation. In

Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services (pp. 73–82).

Shukla, A., Katt, B., Nweke, L. O., Yeng, P. K., & Weldehawaryat, G. K. (2021). System security assurance: A systematic literature review. *arXiv Preprint* arXiv:2110.01904.

Spears, J. L., Barki, H., & Barton, R. R. (2013). Theorizing the concept and role of assurance in information systems security. *Information & Management*, *50*(7), 598–605. https://doi.org/10.1016/j.im.2013.08.004

The Apache Software Foundation (2011). *Apache Jena*. Retrieved January 30, 2022, from https://jena.apache.org/

Tsoumas, B., & Gritzalis, D. (2006). Towards an ontology-based security management. In *20th International Conference on Advanced Information Networking and Applications, 2006. AINA 2006* (pp. 985–992). IEEE.

Tudorache, T., Nyulas, C., Noy, N. F., & Musen, M. A. (2013). WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semantic Web*, *4*(1), 89–99. https://doi.org/10.3233/SW-2012-0057

Villagrán-Velasco, O., Fernández, E. B., & Ortega-Arjona, J. (2020). Refining the evaluation of the degree of security of a system built using security patterns. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (pp. 1–7).

W3C. (n.d.). *RDF 1.1 XML syntax*. Retrieved January 26, 2022, from https://www.w3.org/TR/rdf-syntax-grammar/

W3C. (2004). *SWRL: A semantic web rule language combining OWL and RuleML*. Retrieved June 3, 2022, from https://www.w3.org/Submission/SWRL/

W3C. (2012). *Web ontology language (OWL)*. Retrieved June 3, 2022, from https://www.w3.org/OWL/

W3C. (2013a). *SPARQL 1.1 query language*. Retrieved June 3, 2022, from https://www.w3.org/TR/sparql11-query/.

W3C. (2013b). *W3C semantic web activity*. Retrieved June 3, 2022, from https://www.w3.org/2001/sw/

Waddell, W., Smith, D., Shufelt, J., & Caton, J. (2011). *Cyberspace operations: What senior leaders need to know about cyberspace*. Army War College, Carlisle Barracks, Center For Strategic Leadership.

Wand, Y., Storey, V. C., & Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems*, *24*(4), 494–528. https://doi.org/10.1145/331983.331989

Wang, J. A., & Guo, M. (2009). OVM: an ontology for vulnerability management. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies* (pp. 1–4).

Wang, J. An, Guo, M., Wang, H., Xia, M.., & L., Zhou. (2009). Ontology-based security assessment for software products. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies* (pp. 1–4).

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. *Journal of Cybersecurity and Privacy*, *2*(3), 587–605. https://doi.org/10.3390/jcp2030030

Yavagal, D. S., Lee, S. W., Ahn, G.-J., & Gandhi, R. A. (2005). Common criteria requirements modeling and its uses for quality of information assurance (QoIA). In *Proceedings of the 43rd Annual Southeast Regional conference-Volume 2* (pp. 130–135).

Zhao, W., & Liu, J. K. (2008). OWL/SWRL representation methodology for EXPRESS-driven product information model: Part II: Practice. *Computers in Industry*, *59*(6), 590–600. https://doi.org/10.1016/j.compind.2008.02.004

Zhou, C., & Ramacciotti, S. (2011). Common criteria: Its limitations and advice on improvement. *Information Systems Security Association ISSA Journal*, *2011*, 24–28.