A Method for Detecting Common Weaknesses in Self-Sovereign Identity Systems Using Domain-Specific Models and Knowledge Graph

Charnon Pattiyanon, Toshiaki Aoki and Daisuke Ishii
Japan Advanced Institute of Science and Technology, Ishikawa, Japan

Keywords: Self-Sovereign Identity, Weakness Detection, Domain-Specific Modeling Language, Knowledge Graph.

Abstract: A Self-Sovereign Identity (SSI) system is a decentralized identity management system based on claims that leverages blockchain technology to empower individuals to manage their personal information and autonomously authenticate services. The SSI system is unique in that it makes use of disparate terminologies, making analysis arduous and challenging for security specialists. Weakness analysis is a well-known security assurance technique for determining the presence of weaknesses in a target system. Weakness analysis is crucial to the deployment of the SSI system in that it can earn user trust and be verified secure if the majority of detected weaknesses are addressed properly. We seek to leverage domain experience in this work to lessen the effort required to analyze weaknesses by security specialists unfamiliar with the SSI system. This paper presents two domain-specific modeling languages (DSMLs) based on the unified modeling language (UML) communication diagram for embedding domain knowledge about the SSI system and common weaknesses. Then, with the assistance of domain knowledge graphs, this paper presents a method for detecting weaknesses in the links between the two models created by the proposed DSMLs. Precision and accuracy metrics are used to determine the proposed method's performance.

1 INTRODUCTION

A self-sovereign identity (SSI) is a claim-based, decentralized identity management system. It empowers users to manage and control their personally identifiable information (PII) without the intervention of a central authority (e.g., a service provider). It is a new standard for protecting personally identifiable information (PII) in modern software systems, however it is not generally adopted at the moment due to its complexity and concerns about its security and privacy.

To obtain user trust, the SSI system's security has to be taken into consideration because it's closely linked to the manipulation of PII. However, the SSI system's unique terminologies and mechanism make it more difficult for security specialists to become acquainted. Due to a lack of acquaintance and domain experience, it may result in cases being overlooked when performing security analysis.

Weakness analysis and detection are two frequently used security assurance approaches for enhancing the security of software systems. The ultimate goal is to identify and eliminate any security weaknesses in a target system. MITRE Corporation assists security analysts by compiling a public database, i.e., the common weakness enumeration (CWE), of common software and hardware weaknesses. Several previous works attempted to provide a variety of methodologies for detecting common CWE weaknesses, including an ontological approach (Ansarinia et al., 2012; Salahi and Ansarinia, 2013) and a program analysis approach (Sun et al., 2014). Unfortunately, none of them consider domain knowledge and require domain expertise to implement. Existing methodologies are difficult for inexperience analysts to apply to the SSI system.

Due to the cutting-edge nature of the SSI system and its implementation's emphasis on adhering to its fundamental concepts, its implementation may introduce several weaknesses unintentionally. Thus, weakness analysis becomes critical to the SSI system's security in the sense that it can only be proven secure if the majority of detected weaknesses are adequately addressed. For instance, a weakness in the cleanup of sensitive data following processing. The fundamental concepts made no reference to or concern themselves with data cleansing, resulting in the retention of sensitive information in the memory. Without conducting a weakness analysis, this weakness may become a point of attack for any threat actor.

Pattiyanon, C., Aoki, T. and Ishii, D.
A Method for Detecting Common Weaknesses in Self-Sovereign Identity Systems Using Domain-Specific Models and Knowledge Graph.
DOI: 10.5220/0010824900003119
In *Proceedings of the 10th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2022)*, pages 219-226
ISBN: 978-989-758-550-0; ISSN: 2184-4348

The purpose of this study is to exploit the SSI system's domain expertise and to bridge gaps in commonly used CWE terms. We profile the unified modeling language (UML) communication diagram in order to propose two domain-specific modeling languages (DSMLs): one for the SSI system and another for the CWE weaknesses. Then, we conduct a domain analysis to develop linkages between the SSI system's unique terminologies and the CWE's common weaknesses, which we formalize as a knowledge graph. Finally, we present a methodology for detecting the CWE's common weakness on the basis of the knowledge graph. We undertake an experiment to determine the performance of the proposed method using experts' ground truth. The main contributions of this paper are listed as:

- We abstract the SSI system's native properties using a DSML that is both versatile and comprehensive for modeling the SSI system's architecture.
- We abstract the description of the CWE's common weaknesses into a DSML that can be applied to a variety of areas and methodologies.
- We describe an approach for bridging gaps between standard software development and domain-specific system development. This method enables security specialists who are unfamiliar with the SSI system to do a coverage weakness analysis.

The remainder of this paper is structured as follows: Section 2 discusses the approaches employed. Section 3 proposes a meta-model for DSMLs. Section 4 details our proposed method for detecting weaknesses. Sections 5 and 6 summarize our experimental findings and discuss them. Section 7 compares our work to related works, and Section 8 summarizes our findings and suggests future directions.

2 BACKGROUND

2.1 Self-Sovereign Identity Systems

A self-sovereign identity was invented as an identity-as-a-service solution that overcomes management difficulties through the decentralization of blockchain technology. It guarantees users' complete sovereignty over their PII and their ability to selectively disclose it as little as necessary (Allen, 2016). The SSI system is built on the W3C standard for verifiable credential data models (Sporny et al., 2019), which specifies *three* roles: holder, issuer, and verifier. A **holder** is an individual who is an identity subject and who owns PIIs. An **issuer** is a person, organization, or system that is responsible for validating claims. A

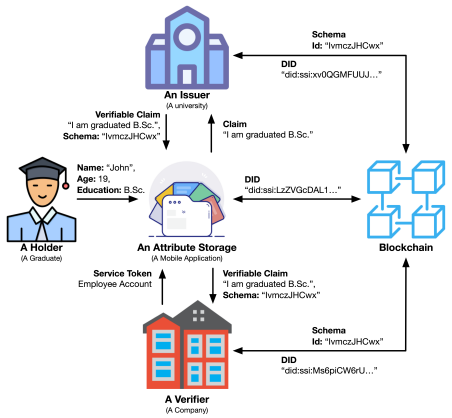

Figure 1: A high-level overview of the SSI system with an actual situation.

verifier is a person, organization, or system that verifies claims to ensure they are valid for service authentication. Mühle et al. (2018) surveyed and reported four essential components: identification, attribute storage, verifiable claim, and authentication. Each entity in the SSI system is uniquely *identifiable* using a decentralized identifier (**DID**). An entity can exchange a DID in order to retrieve the public key of another entity from a blockchain. A holder could store PIIs in the *attribute storage* of a local device and create a claim attesting to a PII with little knowledge. For instance, a holder may record his or her age and create a claim attesting to the fact that the holder is over the age of 18 without disclosing the actual age. The claim should be sent to the issuer for verification of its accuracy. If the claim is valid, the issuer can verify it manually or against existing data and then generate a *verifiable claim* using a blockchain-published schema. To *authenticate* services, a holder may offer both claims and verified claims to a verifier. The verifier may validate the verifiable claim by accessing the related schema on blockchain.

To summarize and explain an actual example, Fig. 1 provides a high-level overview of the SSI system for a graduate who claims a degree in order to seek for a job. The university that grants the degree will publish a degree schema on blockchain, which the company may validate.

2.2 Common Weakness Enumeration

A weakness is a type of fault that can be exploited as a vulnerability within a software product even when it is operating normally (MITRE, 2006). MITRE Corporation urges that the IT community submit common software and hardware weaknesses to the CWE database. They will publish proposed weaknesses in the CWE database after a thorough evaluation.

Multiple text fields are used to record an entry in the CWE database, including a description, common consequences, and demonstrative examples. To determine the occurrence of a weakness, analysts must scan through those text fields and justify the potential in the target system. Only a few entries are provided in the demonstrative example text field as an example of smelly source code. It can be seen that the most important information for detecting a weakness is the description text field, e.g.:

> *"The **software** does not **perform** or incorrectly performs an authorization check when an actor attempts to **access a resource** or **perform an action**."*

Discussing the contents of the description text field, it describes how a system actor acts on information in a positive or negative manner. We focus on the description text field of the common *software weakness* solely because the SSI system's unique functionalities are based on its software application.

2.3 UML Communication Diagram

The UML communication diagram is a subset of the interaction diagram specified by the object management group (OMG). Hundreds of details were given in the specification for each option of the interaction diagrams (Object Management Group, 2017). They did, however, share the same abstract syntax, even though the communication diagram may not incorporate all of them.

Three significant syntaxes demand special attention: a lifeline, a message, and a general ordering. A **lifeline** is a timeline that depicts the progression of a procedure. A **message** is merely the record of events occurring between two lifelines. It will be denoted by a message occurrence specification, which specifies the event's name, arguments, and value. A general ordering denotes the sequence in which the messages appear in the diagram.

In this work, we profile the domain knowledge of the SSI system and the CWE weaknesses that define two novel DSMLs using the abstract syntax of the UML interaction diagram, particularly the section used by the communication diagram.

2.4 Knowledge Graph

A knowledge graph is a directed labelled graph that depicts the relationships between knowledge entities. Ji et al. (2021) recently defined a knowledge graph $\mathcal{G} = \langle \mathcal{E}, \mathcal{R}, \mathcal{F} \rangle$ in which $\mathcal{E}$ denotes a set of entities, $\mathcal{R}$ denotes a set of relations, and $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ denotes a set of facts showing the existence of a relationship between two entities. For example:

- ("John", *born_in*, "Sydney") is a fact denoting john was born in Sydney.
- ("Application", *read*, "data") is a fact denoting an application reads data.

The knowledge graph is a straightforward model that facilitates both user accessibility and systematic search. While it lacks the richness of other graph representations (e.g., ontology), it is lightweight and suffices to denote simple relationships. We employ a knowledge graph to depict the links between SSI system-specific and common software concepts in this work, based on the domain analysis results.

3 META-MODELS

3.1 SSI System DSML Meta-model

Due to the fact that the SSI system is defined by distinct terminologies, and these terminology may vary among publications, designing the SSI system using the UML interaction diagram may require a generic reference to domain-specific information.

According to (Allen, 2016; Sporny et al., 2019; Mühle et al., 2018), the SSI system consists of three interchangeable roles, distinct data objects, and technical components, as stated in Section 1. Moreover, Ferdous et al. (2019) studied the fundamental concept of the SSI system in the literature and identified four critical data objects in formal terms, i.e., a partial identity, an SSI, an assertion, and a profile. Likewise, other works (Tobin and Reed, 2017; Haddouti and Ech-Cherif El Kettani, 2019; Liu et al., 2020; Panait et al., 2020; Wang and De Filippi, 2020; Naik and Jenkins, 2020) studied and presented designs for the SSI system in a variety of ways. Taking this literature into account, we found a common structure of the SSI system's functionalities, i.e., technical component(s) performs a function on SSI-related object(s). Thus, we summarize variations of element in the common structure in Table 1.

We use the common structure as a core notion to create a DSML for the SSI system, as illustrated in Fig. 2 on the left as a meta-model with a class diagram. The meta-model demonstrates how the SSI system's DSML is profiled using the abstract syntax of the UML interaction diagram. The technical components of the SSI system provide a lifeline. Then, functions and SSI-related objects are used to construct the message and value specification, respectively. In this work, we will model the SSI system using the

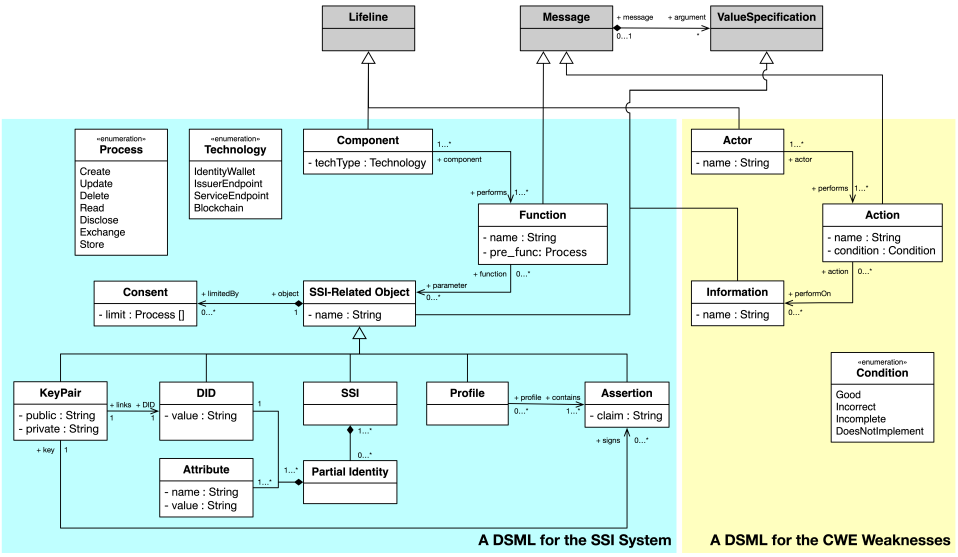

Figure 2: Meta-models of the two DSMLs for the SSI system (left) and the CWE weaknesses (right).

Table 1: Common structure of the SSI system's functions and its variations.

Element	Variation
Technical Component	Holder, Issuer, Verifier, Identity Wallet, Service Endpoint, Blockchain
Function	Create, Update, Delete, Modify, Read, Disclose, Exchange, Store
SSI-related Objects	PII, Identity Attribute, Attribute Value, DID, DID Document, Partial Identity, SSI, Claim, Attestation, Assertion, Proof, Service Token, Public Key, Private Key, Claim Schema, Request Schema, Consent

DSML and then use the model as an input to the proposed weakness detection method.

3.2 CWE Weakness DSML Meta-model

As stated in Section 2.2, the CWE contains hundreds of entries for common software and hardware weaknesses. However, important data is contained in the description text field for the purpose of detecting the occurrence of the weakness.

We divide the important data necessary for detecting the occurrence of weakness into three categories: actors who produce the weakness; actions denote connected actions that, in some conditions, result in a weakness; information refers to a set of data elements that are exploited by an action of weakness. Then, as illustrated on the right-hand side of Fig. 2, we profile the UML interaction diagram to highlight those categories and propose a DSML for the CWE weaknesses. In this work, we will use the proposed DSML to model each entry of the CWE weaknesses and then use it as another input to the proposed method.

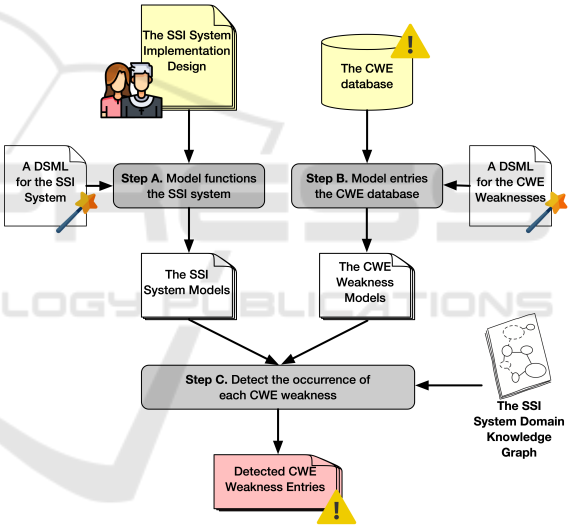

Figure 3: An overview of the knowledge graph-based weakness detection method.

4 KNOWLEDGE GRAPH-BASED WEAKNESS DETECTION

This section describes our proposed method for detecting security software weaknesses in the CWE database by leveraging the inter-model links between the two DSMLs and the predefined SSI system domain knowledge graph. We shall illustrate an overview of the proposed method in Fig. 3.

The proposed method is built on the assumption that a generic description of the CWE weaknesses is adequate to compare to the SSI system's unique

functionalities and detect the existence. At step A, an analyst should model the target SSI system implementation using the proposed DSML for the SSI system. The resulting system model will be a UML communication diagram enhanced with SSI-specific stereotypes. Then, in step B, an analyst should use the proposed DSML to model all the interesting entries of common software weaknesses from the CWE database. Models produced should follow the same format as the SSI system design. Finally, at step C, we propose the SSI system domain knowledge graph based on the results of the domain analysis, which enables an analyst to make use of preexisting information when detecting which weaknesses occur in the target SSI system implementation. In the following sub-section, we will explain each step in details.

4.1 Detailed Methodology

Step A. Modeling of the SSI System Design. This step will use the proposed DSML to model the SSI system. The primary user of this methodology is an analyst who is tasked with analyzing the security of an SSI system implementation. First, regardless of the design process used, an analyst should extract the implementation's functionality. It is sufficient to generate a list of functions, each of which describes which components or actors perform an action on a set of data objects. Then, using the DSML meta-model, an analyst should model those functions as a series of UML communication diagrams with their constituents labeled with the SSI system-specific stereotype (on the left-hand side of Fig. 2). It is possible for an analyst to create many diagrams to represent all of the implementation's unique functionalities. An example of a modeled communication diagram is shown in Fig. 4. Additionally, one can observe that certain additional notations have been added to the communication diagram to clarify the particular stereotype.

Step B. Modeling of the CWE Weaknesses. This step will use the proposed DSML to model the CWE database's weaknesses in the same way as the SSI system does. At the start of this step, an analyst must determine the extent of software weaknesses to examine. While it is ideal to model and use all software weaknesses in the CWE database, this can be scoped to conserve resources and effort. Then, an analyst should utilize the DSML meta-model to represent each software weakness entry as a detailed communication diagram, as this may contain sufficient information to detect a weakness (according to the right-hand side of Fig. 2). A single entry for a software weakness should have a single matching model. An example of a modeled communication diagram is shown in Fig. 5. Specified weakness conditions are highlighted in red texts.

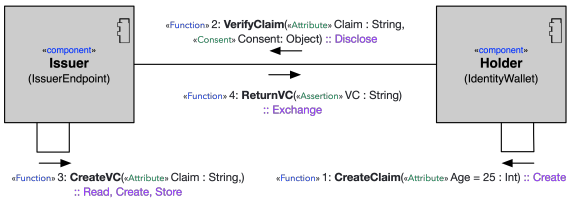

Figure 4: An example of a communication diagram specified by the proposed DSML for the SSI system.

Figure 5: An example of the CWE 212 communication diagram specified by the proposed DSML for the CWE weaknesses.

Step C. Detecting of the Occurrence of Each CWE Weakness. This step will examine two groups of communication diagrams and identify the inter-model links between them. Regrettably, the two groups are frequently represented in disparate circumstances, particularly in terms of nomenclature.

To overcome this issue, we propose a knowledge graph including pre-defined links between the SSI system's specific and common software terms. We analyzed a collection of research papers on the design of SSI systems and discovered that, even within this discipline, distinct names were employed to denote the same meaning. For instance, the terms "user" and "holder" are used interchangeably to refer to a holder. In accordance with (Ji et al., 2021)'s definition, we limit the scope of relations $\mathcal{R}$ in the proposed knowledge graph to the equivalence-to relation (abbreviated "eqv_to"). In Fig. 6, we define links between SSI system-specific terms that share a common meaning through the use of blue relational edges.

On the other hand, we also found that the SSI system-specific terms are somewhat different to common software development terms. We take the specific terms to search in a thesaurus corpora[1] and include synonyms of the specific terms into the knowledge graph. We define links between SSI system-specific terms to common terms using orange relations edges in Fig. 6.

This step detects the occurrence of a weakness in the target SSI system implementation using the

[1] An online thesaurus database: www.thesaurus.com

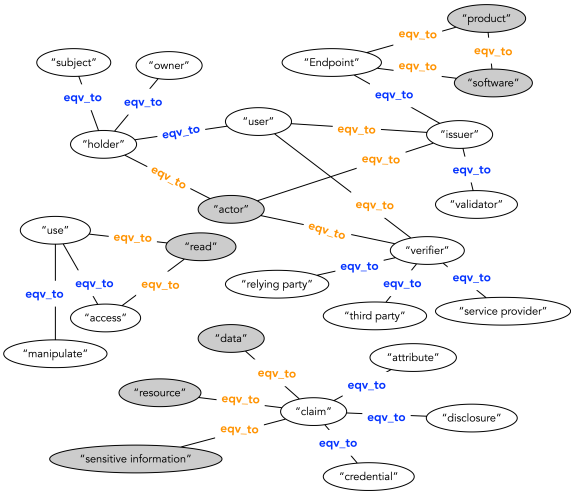

Figure 6: The proposed knowledge containing pre-defined linkages of terms.

knowledge graph and a series of methods outlined in Algorithm 1. This algorithm indicates that all elements of two communication diagrams are iterated and compared using a procedure called CHECKEQUIVALENT(). The procedure should verify if a pair of elements are present in the knowledge graph and have relevant relations. If this is the case, the pair will be collected for the purpose of indicating the occurrence of weakness. Otherwise, a manual judgment will be required to update the knowledge graph.

Algorithm 1: Weakness detection between communication diagrams for the SSI system and the CWE weakness using the knowledge graph.

Input: A communication diagram of the SSI system (S)
A communication diagram for a weakness (W)
A knowledge graph ($\mathcal{G}_{SSI}$)
Output: A boolean decision of the weakness occurrence.

Initiate $Rel_c = False, Rel_f = False, Rel_i = False$;
for each component c in S **do**
  for each actor a in W **do**
    if CHECKEQUIVELENT(c,a) **then**
      $Rel_c = True$;
for each function f in S **do**
  for each action p in W **do**
    if CHECKEQUIVALENT(f,p,G_{SSI}) **then**
      Check conditions and $Rel_f = True$;
for each SSI-related object or its inheritance o in S **do**
  for each information i in W **do**
    if CHECKEQUIVALENT(o,i,G_{SSI}) **then**
      $Rel_i = True$;
return Rel_c **and** Rel_f **and** Rel_i;
procedure CHECKEQUIVALENT(t_1,t_2,G_{SSI})
  if $(t_1$, `eqv_to`, $t_2)$ in $\mathcal{F}$ of $\mathcal{G}_{SSI}$ **then**
    return True;
  else
    return Determine whether t_1 and t_2 are equivalent and update $\mathcal{G}_{SSI}$;

Table 2: Statistical data of the experimental runs.

Product	#SSI	#CWE	#Pair	#Rel	#Detect	Time
IBM	5	10	18,363	9	3	1.49s
Sovrin	7	10	20,528	15	3	1.67s
uPort	2	10	10,551	13	3	1.32s

Fig. 4 and 5 illustrate the proposed method's results in action. Assume we model a function representing an implementation in Fig. 4 and the CWE 212 entry in Fig. 5 using steps A and B. According to the CWE 212 weakness, the product does not remove or only partially remove sensitive information prior to transfer, allowing other users to access it. Using the knowledge graph, Algorithm 1 determines if an issuer endpoint is equivalent to a product (see at the top-right of Fig. 6). Additionally, the claim is equivalent to sensitive information. The discovered relationships appear to suggest that there is a possibility of CWE 212 entry in the SSI system implementation.

5 EVALUATION

5.1 Implementation

We develop a Python command-line interface tool to aid in the implementation of the proposed method in real-world scenarios. The program accepts two JSON files as inputs: one containing the SSI system model and another containing an entry for a CWE weakness. The JSON keys are defined using the DSMLs and are structured according to the UML communication diagram's core elements. The tool is then used to conduct Algorithm 1 semi-automatically, as some aspects still require manual judgment. The knowledge graph, on the other hand, is designed to be evolutionary, and its updating will bring the tool closer to automaticity.

5.2 Experiment Settings and Findings

We conducted an experiment to determine the SSI system's performance in real-world implementations. We chose three implementations of the SSI system as examples: IBM Verify Credentials, Sovrin, and uPort. The IBM Verify Credentials and the Sovrin have a similar target market (i.e., enterprise-level businesses), but their network governance mechanisms are distinct. uPort, on the other hand, is more compact that includes a variety of features. They are excellent representations of real-world differences.

In JSON files, we encode the functions of the three implementations and 10 CWE weakness entries. The 10 entries were chosen at random from a bucket of software weaknesses with adequate de-

Table 3: An evaluation result of the proposed method.

Product	Confusion Matrix				Accuracy	Precision
	TP	FP	FN	TN		
IBM	3	0	1	6	90.00%	100.00%
Sovrin	3	0	0	7	100.00%	100.00%
uPort	2	1	2	5	70.00%	66.67%

scriptions to enable detection of their occurrence. The JSON files are then supplied into the implemented tool, which detects the occurrence of weaknesses. Table 2 summarizes statistical data from experimental runs in which three implementations were compared to 10 entries (#CWE). The implementation is encoded as a series of communication diagrams (#SSI), each of which contains a significant number of elements to compare against the CWE weakness entries. The fourth column (#Pair) denotes the number of iterated term pairs in Algorithm 1, while the fifth and sixth columns (#Rel and #Detect) denote the number of discovered relations and detected weakness entries, respectively. The final column indicates the time required for the implemented tool to complete the detection. It should be noted that the time is only recorded for the automatic phase, and the time spent on manual judgment is not included.

We invited three specialists with at least three years of expertise in the field of identity management, security training, and a sufficient level of English reading ability to manually identify instances of weakness. Expert responses serve as the ground truth for the evaluation, in which we assess the proposed method's performance using the confusion matrix. It has four possible values: true positive (TP) indicates an accurate detection that corresponds to expert opinion; false positive (FP) indicates an incorrect detection of the occurrence; false negative (FN) indicates an incorrect detection of the non-occurrence; and true negative (TN) indicates an accurate detection of a weakness that will not occur. The results of each implementation are shown in Table 3, along with computed precision and accuracy measures.

6 DISCUSSION

Performance of the Proposed Method. Due to the fact that the two domains are defined in distinct contexts, the proposed method's performance is characterized by its ability to close gaps and identify the occurrence of weaknesses. Two performance advantages are clear from the experimental results. To begin, the accuracy and precision measures indicate that the information retrieval performance is satisfactory. Indeed, accuracy is greater than 70% and precision is greater than 65% in three real-world scenarios. Although the accuracy and precision ratings are unsurprisingly high, analysts unfamiliar with the SSI system may have a decent starting point for narrowing the search scope for CWE database weaknesses. Second, the proposed method makes advantage of an evolutionary knowledge graph. It may be updated on a regular basis, and its performance could be improved over time. It is advantageous that the implemented tool runs in less than two seconds, which is faster than manual scanning.

Two limits in terms of faulty cases were observed. In Table 3, we identified instances of false positives, indicating a deceptive detection. The cases may result from the procedure's imperfect knowledge graph. However, due to its evolutionary nature, it has the potential to transcend this constraint over time. The limitation is the potential of false negative situations due to a lack of semantic continuity. For instance, certain CWE weaknesses necessitate human judgment that goes beyond the concept of links. This constraint implies that the proposed method will continue to require human engagement.

Extensibility in the Domain. Although the proposed method is designated to be domain-specific, it still provides opportunity to different variations. It shows in the experiment that the proposed method is applicable to different representatives of the SSI system's implementation. If there is an implementation that does not align with the fundamental notion of the SSI system, the proposed method and DSMLs may be limited and require further modification.

Threat to Validity. This work identifies two threats to validity. First, the basis for expert opinions must be reliable. Even if the invited specialists work in the field of identity management, they are not involved with the SSI system. We believe that their experience in that field is adequate for them to comprehend the notion of the SSI system. Second, the sample size of weakness entries is quite small. Given the human factor, the sample size should be modest enough to avoid human error. While the sample size is modest, the size of each entry is reasonable, as more than 10,000 pairs are iterated.

7 RELATED WORK

There are several works attempted to propose weakness detection method in the past few year. An ontological approach is a prominent way to represent the weakness and detect its presence as in used in

(Ansarinia et al., 2012; Salahi and Ansarinia, 2013), but the existing ontologies formulate only the domain of the CWE weakness only. It still requires domain knowledge to understand the target system. Another approach is the use of program analysis, e.g., Sun et al. (2014) and Son et al. (2015), but not all entries contain code examples or resources to perform program analysis. Most existing techniques did not take the domain knowledge of the target into account.

For the proposals for domain-specific models on security purposes, there are some works that attempted to use DSMLs to capture the security characteristics, e.g., security concerns (Silva Gallino et al., 2012), security objectives (Saleem et al., 2012), or attack surfaces (Sun et al., 2020). To the best of our knowledge, none of work has attempted to model the domain-specific knowledge of the two domains.

8 CONCLUSIONS

This paper proposes two DSMLs for profiling domain-specific knowledge of the SSI system's and the CWE weaknesses. We also propose a method for detecting common software weaknesses in a targeting implementation of the SSI system. The proposed method compares two system models utilizing the knowledge graph. We implement a command-line interface tool to semi-automatically process the proposed method and conduct an experiment for evaluating its performance. The proposed method achieves a certain performance that is acceptable.

However, the proposed method obscures some of the knowledge graph's inadequacies and semantic continuity. We believe that the domain knowledge examined and applied in this study might facilitate future research aimed at eliminating those issues.

REFERENCES

Allen, C. (2016). The path to self-sovereign identity. http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

Ansarinia, M., Asghari, S. A., Souzani, A., and Ghaznavi, A. (2012). Ontology-based modeling of ddos attacks for attack plan detection. In *IST 2012*, pages 993–998.

Ferdous, M. S., Chowdhury, F., and Alassafi, M. O. (2019). In search of self-sovereign identity leveraging blockchain technology. *IEEE Access*, 7:103059–103079.

Haddouti, S. E. and Ech-Cherif El Kettani, M. D. (2019). Analysis of identity management systems using blockchain technology. In *CommNet 2019*, pages 1–7.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21.

Liu, Y., He, D., Obaidat, M. S., Kumar, N., Khan, M. K., and Raymond Choo, K. K. (2020). Blockchain-based identity management systems: A review. *Journal of Network and Computer Applications*, 166(February):102731.

MITRE (2006). Common Weakness Enumeration (CWE). https://cwe.mitre.org/.

Mühle, A., Grüner, A., Gayvoronskaya, T., and Meinel, C. (2018). A survey on essential components of a self-sovereign identity. *Computer Science Review*, 30:80–86.

Naik, N. and Jenkins, P. (2020). Governing principles of self-sovereign identity applied to blockchain enabled privacy preserving identity management systems. In *ISSE 2020*, pages 1–6.

Object Management Group (2017). Unified Modeling Language: Specification. Version 2.5.1. formal/17-12-05.

Panait, A.-E., Olimid, R. F., and Stefanescu, A. (2020). Analysis of uport open, an identity management blockchain-based solution. In *TrustBus 2020*, pages 3–13. Springer International Publishing.

Salahi, A. and Ansarinia, M. (2013). Predicting network attacks using ontology-driven inference. http://arxiv.org/abs/1304.0913.

Saleem, M. Q., Jaafar, J. B., and Hassan, M. F. (2012). A domain-specific language for modelling security objectives in a business process models of SOA applications. *Advances in Information Sciences and Service Sciences*, 4(1):353–362.

Silva Gallino, J. P., De Miguel, M., Briones, J. F., and Alonso, A. (2012). Domain-specific multi-modeling of security concerns in service-oriented architectures. *Lecture Notes in Computer Science*, 7176:128–142.

Son, Y., Lee, Y., and Oh, S. (2015). A Software Weakness Analysis Technique for Secure Software. *Advanced Science and Technology Letters*, 93:5–8.

Sporny, M., Longley, D., and Chadwick, D. (2019). Verifiable credential data model v1.0. https://www.w3.org/TR/vc-data-model/.

Sun, F., Xu, L., and Su, Z. (2014). Detecting logic vulnerabilities in e-commerce applications. In *DNSS 2014*, pages 23–26.

Sun, T., Drouot, B., Golra, F., Champeau, J., and Guerin, S. (2020). A Domain-specific Modeling Framework for Attack Surface Modeling. In *ICISSP 2020*, pages 341–348.

Tobin, A. and Reed, D. (2017). The inevitable rise of self-sovereign identity: A white paper from the sovrin foundation. https://sovrin.org/wp-content/uploads/2017/06/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf.

Wang, F. and De Filippi, P. (2020). Self-sovereign identity in a globalized world: Credentials-based identity systems as a driver for economic inclusion. *Frontiers in Blockchain*, 2(January):1–22.