A Comparison Study of Available Software Security Ontologies

Mamdouh Alenezi
College of Computer and Information Sciences
Prince Sultan University
Riyadh, Saudi Arabia
malenezi@psu.edu.sa

Hamid Abdul Basit
College of Computer and Information Sciences
Prince Sultan University
Riyadh, Saudi Arabia
hbasit@psu.edu.sa

Faraz Idris Khan
Security Engineering Lab (SEL)
Prince Sultan University
Riyadh, Saudi Arabia
fikhan@psu.edu.sa

Maham Anwar Beg
Independent Researcher
Lahore, Pakistan
maham.a.beg@gmail.com

ABSTRACT

A rising number of software and services malfunctioning due to security flaws has increased the importance of software security and resulted in numerous knowledge sources of the domain. Building secure software systems require the understanding and extraction of the available knowledge, and a standard knowledge management platform is needed. Ontologies form an integral part of knowledge management platforms as they capture and structure the given knowledge. Various software security ontologies have been proposed previously, either stand-alone or as part of some bigger ontology like a computer or information security. However, these ontologies do not cover the entire domain and cannot be used as a standard ontology for software security in its current form. In this paper, we have identified and evaluated the existing ontologies that specifically capture software security knowledge, both qualitatively and quantitatively with the help of ontology evaluation tools, in order to select the best ontology that can be extended to prepare the standard ontology for the software security domain.

CCS CONCEPTS

• **Software Security → Security Ontology**; *Software Security Knowledge Management*; Software Quality Assurance; Context based software security ontology.

KEYWORDS

Software Security, Software Security Knowledge Management, Security Ontology, Software Quality Assurance, Software Quality Management

ACM Reference Format:
Mamdouh Alenezi, Hamid Abdul Basit, Faraz Idris Khan, and Maham Anwar Beg. 2020. A Comparison Study of Available Software Security Ontologies. In *Evaluation and Assessment in Software Engineering (EASE 2020), April 15–17, 2020, Trondheim, Norway*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3383219.3383292

1 INTRODUCTION

The recent explosion of internet-based applications and services has increased reliance of businesses over them. Internet-based applications and services are frequently attacked by hackers or malicious users. Such attacks result in malfunctioning of the application or service [2]. Also, most of the time the attack affects the availability of the application or service. There are two ways such attacks can be tackled, one is after the attack has occurred which is a reactive approach [16]. The other way is to engineer security into the software, whether its an application or service, which is a preventive approach. Hence, we see an emerging field of software security where researchers are struggling to engineer security into the software being developed. The struggle has led to novel innovations that improve software engineers' capability to develop secure software and protecting the applications. Software security is improved by equipping software engineers with the knowledge and skills to ensure security in the software development life cycle. As a result security attacks can be resisted and security errors can be handled efficiently. Security committees and a group of experts identify vulnerability patterns i.e. CVE (Common Vulnerabilities Exposure) [1] and CWE (Common Weakness Enumeration) [2] managed by MITRE cooperation [4]. CVE is the list of entries where each entry contains an identification number, description and one public reference for publicly known vulnerabilities. CWE is a list developed by a community of common software weaknesses. It acts as a common language and baseline for weakness identification, mitigation and prevention efforts. It also acts as a measuring ruler for software security tools. Apart from these, standards and guidelines for secure software development have been developed, which accumulates the knowledge and experiences of various software development organizations. Some of the latest efforts in software security can be found in [1, 3, 20].

Software security knowledge can extensively be found in various sources [12]. Examples of these sources are checklists, standards and best practices in books, literature, and the world wide web. The existing knowledge is not only highly complex but also very

[1]https://cve.mitre.org//
[2]https://cwe.mitre.org/

context-specific. Such heterogeneous and a wide variety of sources make it difficult for a software development team to gather all the knowledge and apply it correctly in their own application-specific situation during SDLC.

Most recently [8] the above-mentioned problem have motivated researchers to organize software security knowledge. The knowledge is organized in a manner that will assist the software development team to relate the security knowledge in an application-specific situation. To effectively utilize the software security knowledge and make it practically feasible in software development practices, knowledge management is considered the best fit. Within knowledge management, ontology is considered the best approach to categorically organize the domain knowledge of information and software security. Security concepts which include attacks, vulnerabilities, and prevention mechanisms are systematically classified. Not only security information is classified and categorized, but additional contextual features are also incorporated. Such efforts help to capture contextualized security knowledge within the ontology [8].

A number of software security ontologies can be found in literature [8, 10, 13, 21, 28–30, 32]. The objective of developing these ontologies is to provide support for software developers and knowledge users. This will help them to utilize the security knowledge and appropriately apply in a current working context. The ontologies capture the context of the application and domain knowledge regarding security, and provides contextualized knowledge. This helps the knowledge users to perform contextual inquiry through heterogeneous software use cases. Until now in literature no comparison and analysis of available software security ontologies can be found. Motivated with this fact, in this paper we compare and analyze the available software security ontologies. By looking at the shortcomings we encounter in the available software security ontologies, we are interested in constructing an improved and comprehensive software security ontology. The contribution of our work is as follows.

- Selecting available software security ontologies in the literature
- Comparing and analyzing the selected software security ontologies

The paper is organized as follows: In section 2, we discuss the related studies. The adopted methodology is presented in Section 3. Section 4 discusses the quantitative evaluation results of selected software security ontologies with the selected tools, while Section 5 gives a qualitative analysis of the ontologies for the purpose at hand. We conclude the paper in section 6.

2 LITERATURE REVIEW

In this section, we discuss the related work in the area of software security, software security knowledge management, and software security ontologies. A comprehensive literature review in the area of software security in open source development can be found in [31]. The authors expressed a need for reconsideration of software security studies as it helps to understand security practices and weaknesses. Motivated with this, a systematic literature review is conducted to extract software security studies.

Software changes are deployed frequently by following DevOps practices. Most recently, these changes are acting as a medium of injecting vulnerabilities in the software. Hence, there is a dire need for best security practices. Rahman and Williams [26] proposed security practices that can be incorporated in the DevOps practices. The software can be made more secure by incorporating security practices within the software development lifecycle. Khaim et al. [14] performed a literature review of security issues in agile software development. These issues arise due to the lack of involvement of security experts in the software development phases. The authors have discussed the roles and responsibilities of a security expert during the development lifecycle.

Software security concerns are not only raised for internet-based applications and software. They are also found in other application domains such as the Internet of things (IoT). Koivu et al. [15] analyzed heterogeneous security solutions for IoT and put forward appropriate techniques to implement them. Software security vulnerabilities find their way into software often during runtime of software application. One such work can be found in [9] where the filers of the patent proposed a runtime analysis framework for identifying software security vulnerabilities.

Ramachandran [27] identified key methods and techniques of software security requirements engineering for developing secure cloud services. Implementing best security practices within software life cycle can be found in [18]. Mohaddes et al. [19] discussed software security challenges and issues in software development lifecycle. Techniques of knowledge management are extensively used for capturing software security knowledge. Due to critical need for security knowledge to be structured, we can find various efforts of managing them. We discuss these efforts below. Such efforts can be found in [25]. Premchaiswadi et al. stressed for managing software security knowledge by software engineering knowledge management techniques. Nunes et al. [23] discussed techniques to include security relevant information within software engineering artifacts. This is supported by a knowledge management environment.

Research can be found in literature [1] where Abu-Taieh et al. proposed a cyber security body of knowledge useful for knowledge management. Modeling software security knowledge can be found in [7]. Jens et al. used ontologies to manage system-specific and security knowledge. An owl based ontology to model security knowledge can be found in [13]. Security information is classified with the help of ontology in [21]. Vorobiev et al. proposed an ontology-driven approach to capture information security knowledge [30]. In another effort, the authors proposed a common body of knowledge for securing software and services. One such work can be found in [28]. Schwittek et al. proposed an ontology-based approach for mapping security requirements to solutions i.e. security patterns in [10]. A comprehensive security ontology called SecAOnto [8] is an OWL-based model used for security assessment.

An ontology extending the vulnerability concepts provided by the National Institute of Standards and Technology (NIST) can be found in [29]. The ontology can be used to reason, manage, and analyze vulnerabilities. A context-based ontology for managing software security knowledge can be found in [32]. An ontology capturing security knowledge from available repositories and specialized vulnerability databases can be found in [5].

3 METHODOLOGY

In this section, we discuss the methodology adopted in this work.

First of all, we searched for the available software security ontologies in the published literature. As software security is part of computer and information security, we looked into those domains as well. The selection criteria for software security ontologies are threefold; the ontology has to be published in a peer-reviewed avenue between 2005 and 2019, the ontology has to be specific to software security, and ontologies that only discuss information or computer security in general are excluded.

Next, we looked for ontology evaluation tools that can give us quantitative results for the selected ontologies. Several tools are available for ontology evaluation with respect to various criteria [11], but our focus is on ontology design and syntactic correctness, so we selected the tools accordingly. The selection criteria for tools are availability, popularity, features, supported input, ease of use, significant insights provided, and ease of installation of the tools.

In addition to collecting and analyzing various quantitative metrics for the selected ontologies from the selected tools, we also manually analyzed the ontologies to find their fitness for purpose at hand.

Table 1: Software Security Ontologies Dataset

SN	Software Security Ontology	Year
1	Herzog et al. [13]	2007
2	Mourad et al. [21]	2008
3	Vorobiev et al. [30]	2010
4	Schwittek et al. [28]	2012
5	Guan et al. [10]	2016
6	Wen et al. [32]	2018
7	Syed et al. [29]	2018
8	De Franco Rosa et al. [8]	2018

From here onward, we will refer to the ontologies with their serial number as give in Table 1.

4 RESULTS AND DISCUSSION

We evaluated the selected ontologies presented in Table 1. We used Protégé [22] to convert the conceptual models presented in Table 1 into OWL ontologies. The evaluation results are presented below.

Table 2: Quality Metrics and Criteria Metrics [17]

Quality Metrics	Criteria Metrics
Accuracy	Attribute Richness Inheritance Richness Relationship Richness Average Depth Max Depth Average Breadth Max Depth Class Inheritance Richness Class Relationship Richness
Understandability	Absolute Leaf Cardinality Absolute Depth Average Depth Maximum Depth Absolute Breadth Average Breadth Maximum Breadth Class Readability
Cohesion	Absolute Root Cardinality Absolute Leaf Cardinality
Computational Efficiency	Tangledness
Conciseness	Average Population Class Richness

4.1 OntoCheck Evaluation Results

OntoCheck[3] is a plugin for one of the most widely used open-source ontology tools Protégé. OntoCheck allows performing clean-up checks on OWL Ontologies. It reports violations and inconsistencies in naming conventions. Naming conventions are crucial in ontology engineering. They allow ontologies to be human and machine-readable, reduce heterogeneity, and ease the process of integration and mapping of OWL ontologies. We evaluate the results of violations in most common naming convention practices using OntoCheck in Table 3. The numbers in Table 3 represent a violation of a naming convention in percentage for a given ontology.

[3]http://www2.imbi.uni-freiburg.de/ontology/OntoCheck

Table 3: OntoCheck Software Security Ontology Evaluation. Results report naming convention violations in percentage.

Naming Convention ‖*Ontologies* :	1	2	3	4	5	6	7	8
All Capital Case	94.4	87.5	92.8	94.4	94.7	95.4	93.9	97
All Lower Case	94.4	87.5	45.2	94.4	94.7	95.4	96.9	26.4
Upper Case Start	27.7	0	2.3	50	57.8	98.9	51.5	26.4
Camel Case	0	0	0	0	5.2	98.9	0	0
Camel Hump	94.4	87.5	95.2	94.4	94.7	96.4	96.9	97

The results in Table 3 shows that most conceptual models uses the camel case naming conventions while ontology by Wen [32] does not follow any naming convention and has high violation percentage in all commonly used naming conventions.

4.2 RDF Triple Checker Evaluation Results

RDF Triple checker [4] identifies common errors found in RDF data. The tool verifies the use of RDF Schema[5] in a given OWL ontology and points out accuracy and errors. RDF tripple checker provides more insight to RDF syntax than OWL Manchester [6]. Evaluation by RDF Triple checker shows that all software security ontologies

[4]http://graphite.ecs.soton.ac.uk/checker/
[5]https://www.w3.org/TR/rdf-schema/
[6]http://visualdataweb.de/validator/

commonly use schema elements *Type*, *Domain*, *Range*, *Ontology*, *objectProperty*, and *Class*. These elements are defined correctly and are indicated as "OK" by tool. Ontologies 3 and 8 use an additional schema element *SubclassOf* which is defined correctly aswell and indicated as "OK". The results indicate that all the ontologies are conceptually valid for further processing of RDF schema based queries.

4.3 Ontology Pitfall Scanner Evaluation Results

OOPS! [7] is a widely used web-based ontology evaluation tool with high availability. OOPS! examines ontologies against 33 common pitfalls. It reports additional pitfalls detected by many existing tools[24].

From the Table 4, we find that Ontologies 3, 4, and 7 have most number of pitfall cases, while Ontology 6 has the least number of pitfall cases. ‘Missing annotations’ is the most common type of pitfall, followed by ‘Inverse relationship not explicitly declared’.

4.4 OntoMetric Evaluation Results

A framework for metrics in the wider area of ontology engineering is provided by web-based ontology evaluation tool OntoMetric[8]. The provided quality metrics help with processes or methodologies for the design of ontologies and in the selection of the best ontology among several. We use OntoMetrics to evaluate to which extent and how good methodologies, practices, and guidelines have been followed by the selected software security ontologies. The tool calculates base metrics, graph metrics, and schema metrics. Base metrics show the number of various ontology elements. Schema metrics assess the design of ontologies, while graph metrics analyze the structure of ontologies. In the base metrics DL expressivity, the expressivity is encoded in the label for logic using AL (Attributive Language) and ALH (Attributive Language Role Hierarchy).

We evaluate the quality of the software security ontologies with the help of the Base Metrics (Table 5), Graph Metrics (Table 6), Schema Metrics (Table 7), Quality and Criteria metrics (Table 2) proposed in [17]. From Table 2, with respect to ontology accuracy and understandability, we find ontologies 3,7, and 8 to be moderately accurate and understandable. Ontologies 1, 2, and 4 are found to be less accurate and understandable. Ontology 6 is found to be relatively better than other ontologies in terms of accuracy and understandability. When looking at computational efficiency, all of the ontologies except 5 were found be to be computationally efficient. Ontologies 4, 7 and 8 have high cohesion, Ontologies 5 and 6 have moderate, while Ontologies 1 and 2 have low cohesion. All of the available software security ontologies were found to be concise.

5 QUALITATIVE ANALYSIS

In this section, we qualitatively analyze the selected ontologies for the purpose of knowledge management of software security domain and comment on their suitability. Ontology 1 is an information security ontology with more generic terms and covers some aspects of software security. However, the terminology used is not

[7]http://oops.linkeddata.es/response.jsp
[8]https://ontometrics.informatik.uni-rostock.de/ontologymetrics/

Table 4: Ontology Pitfall Scanner (OOPS) Checker Software Security Ontology Evaluation

Ontology	Pitfalls	Cases
1	Creating unconnected ontology elements	1
	Missing annotations	32
	Missing disjointness	1
	Inverse relationship not explicitly declared	16
	No License declared	1
2	Creating unconnected ontology elements	12
	Missing annotations	1
	Missing disjointness	6
	Inverse relationship not explicitly declared	1
	No License declared	1
3	Using "is" instead of "rdfs:subClassOf" or "rdf:type" or "owl:sameAs"	1
	Missing annotations	94
	Missing disjointness	1
	Inverse relationship not explicitly declared	55
	Using different naming conventions in ontology	1
	Equivalent classes not explicitly declared	2
	No License declared	1
4	Creating unconnected ontology elements	2
	Missing annotations	47
	Missing disjointness	1
	Inverse relationship not explicitly declared	27
	No License declared	1
5	Missing annotations	37
	Missing disjointness	1
	Inverse relationship not explicitly declared	20
	No license declared	1
6	Creating unconnected ontology elements	2
	Missing annotations	32
	disjointess	1
	Inverse relationship not declared	12
	No License declared	1
7	Creating unconnected ontology elements	1
	Missing annotations	61
	Missing disjointness	1
	Inverse relationship not explicitly declared	30
	No License declared	1
8	Creating unconnected ontology elements	10
	Missing annotations	46
	Missing disjointness	1
	Inverse relationship not explicitly declared	14
	Equivalent classes not explicitly declared	2
	No License declared	1

Table 5: Base Metrics

Metrics ||*Ontologies* :	1	2	3	4	5	6	7	8
Axioms	71	24	203	101	77	112	121	101
Logical Axiom Counts	39	12	109	54	40	48	60	54
Class Count	16	6	40	20	17	20	31	20
Total Classes Count	16	6	40	20	17	20	31	20
Object Property Count	16	6	55	27	20	13	30	27
Total Object Properties Count	16	6	55	27	20	13	30	27
Data Property Count	0	0	0	0	0	0	0	0
Total Data Properties Count	0	0	0	0	0	0	0	0
Properties Count	16	6	55	27	20	13	30	27
Individual Count	0	0	0	0	0	0	0	0
Total Individuals	0	0	0	0	0	0	0	0
DL expressivity	AL	AL	ALH	AL	AL	ALH	AL	AL

Table 6: Graph Metrics

Metrics ||*Ontologies* :	1	2	3	4	5	6	7	8
Absolute root cardinality	9	6	9	20	17	8	40	20
Absolute leaf cardinality	13	6	13	20	17	15	40	20
Absolute sibling cardinality	16	6	16	20	17	20	40	20
Absolute depth	23	6	23	20	17	32	40	20
Average depth	1.4375	1	1.4375	1	1	1.6	1	1
Maximal depth	2	1	2	1	1	2	1	1
Absolute breadth	16	6	16	20	17	20	40	20
Average breadth	4	6	4	20	17	3.33	40	20
Maximal breadth	9	6	9	20	17	8	40	20
Ratio of leaf fan-outness	0.8125	1	0.8125	1	1	0.75	1	1
Ratio of sibling fan-outness	1	1	1	1	1	1	1	1
Tangledness	0	0	0	0	0	0	0	0
Total Number of Paths	16	6	16	20	17	20	40	20
Average Number of Paths	8	6	8	20	17	10	40	20

Table 7: Schema Metrics

Metrics ||*Ontologies* :	1	2	3	4	5	6	7	8
Attribute richness	0	0	0	0	0	0	0	0
Inheritance richness	0.5	0	0	0	0	0.6	0.5625	0
Relationship richness	0.4838	1	1	1	1	0.52	0.4375	1
Attribute class ratio	0	0	0	0	0	0	0	0
Equivalence ratio	0	0	0	0	0	0	0	0
Axiom/Class ratio	2.9	4	5.075	5.05	4.529	5.6	2.875	3.9
Inverse relations ratio	0	0	0	0	0	0	0	0
Class/relation ratio	1.03	1	0.72	0.74	0.85	0.8	1	1.03

enough to cover software security knowledge management. Ontology 2 is focused on the hardening and patching of open-source software systems. It does not address all relevant software development phases. Ontology 3 is a general information security ontology. It is too complicated with lots of terms and layers and only addresses some relevant terms of software security. Ontology 4 is dedicated to a common body of knowledge for engineering secure software and services. The ontology lacks several integral parts of software security knowledge management terms such as attacks, vulnerabilities, and countermeasures. Ontology 5 is more focused on security pattern selection. It helps practitioners to select the appropriate security patterns for a specific situation. Ontology 6 is the only ontology that is specific to software security knowledge management. It is an initial work that is not yet mature. The authors decided to separate the ontology into two contexts - application and domain model. The ontology lacks several main concepts about software security knowledge management. It focuses on the implementation level of the software development lifecycle. Ontology 7 focuses more on vulnerabilities management which is only a subset of software security. Ontology 8 is more focused on security assessments. It covers aspects of how to assess the software system with regard to security. It lacks several integral parts of software security knowledge management terms.

6 THREATS TO VALIDITY

For our study, we took measures to mitigate potential threats to validity which can be internal validity and conclusion validity [6]. The threat to internal validity in this study could lie in the selection bias (i.e. available software security ontologies were selected for analysis). While shortlisting the available software security ontologies, we made sure to include ontologies only available in the software security domain. We excluded the ontologies available in the domain of information security and computer security. We believe the selection bias is mitigated. Conclusion validity deals with the correctness of the conclusion reached in the study. We mitigated the threat to conclusion validity as we used standard and popular tools for evaluating the available ontologies.

7 CONCLUSION

An increasing number of cyber attacks on internet-based applications and services due to software weaknesses demands for software security knowledge management. Efforts have been made to organize security knowledge using knowledge management techniques such as ontologies. Hence, this has resulted in numerous security ontologies in literature. Ontologies for organizing software security knowledge and their analysis is least researched upon. Therefore in this paper, we have identified software security ontologies available in the literature and the tools that can analyze various aspects of these ontologies. We have analyzed and evaluated eight ontologies using four popular tools i.e. OntoCheck, OntoMetrics, OOPS, and RDF Tripple Checker tools. Moreover, we have discussed these results and also provided a qualitative assessment of the selected ontologies. The ontology proposed by Wen [32] was found to be reasonably accurate, cohesive, computationally efficient, and concise. It also has the least number of pitfall cases when evaluated by OOPS. We believe that this ontology can be used as the basis for creating the standard ontology of the software security domain, which will also cover the areas of the domain not yet captured by the existing ontologies. As part of future work, this standard software security ontology will be used to develop the standard software security knowledge management platform for software practitioners.

8 ACKNOWLEDGEMENT

The authors are grateful to Prince Sultan University for supporting this work.

REFERENCES

[1] E. M. O. Abu-Taieh. 2017. Cyber Security Body of Knowledge. In *2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2)*. IEEE, 104–111. https://doi.org/10.1109/SC2.2017.23

[2] Alka Agrawal, Mamdouh Alenezi, Rajeev Kumar, and Raees Ahmad Khan. 2019. A source code perspective framework to produce secure web applications. *Computer Fraud & Security* 2019, 10 (2019), 11–18.

[3] Mamdouh Alenezi and Ibrahim Abunadi. 2015. Evaluating software metrics as predictors of software vulnerabilities. *International Journal of Security and Its Applications* 9, 10 (2015), 231–240.

[4] Mamdouh Alenezi and Yasir Javed. 2016. Developer companion: A framework to produce secure web applications. *International Journal of Computer Science and Information Security* 14, 7 (2016), 12.

[5] Sultan S Alqahtani, Ellis E Eghan, and Juergen Rilling. 2016. Tracing known security vulnerabilities in software repositories–A Semantic Web enabled modeling approach. *Science of Computer Programming* 121 (2016), 153–175.

[6] V. R. Basili, R. W. Selby, and D. H. Hutchens. 1986. Experimentation in software engineering. *IEEE Transactions on Software Engineering* SE-12, 7 (July 1986), 733–743. https://doi.org/10.1109/TSE.1986.6312975

[7] Jens Bürger, Daniel Strüber, Stefan Gärtner, Thomas Ruhroth, Jan Jürjens, and Kurt Schneider. 2018. A framework for semi-automated co-evolution of security knowledge and system models. *Journal of Systems and Software* 139 (2018), 142–160.

[8] Ferrucio de Franco Rosa, Mario Jino, and Rodrigo Bonacin. 2018. Towards an Ontology of Security Assessment: a core model proposal. In *Information Technology-New Generations*. Springer, 75–80.

[9] Sergey Gorbaty, Travis Safford, Xiaoran Wang, and Yoel Gluck. 2018. Runtime analysis of software security vulnerabilities. US Patent 10,140,456.

[10] Hui Guan, Hongji Yang, and Jun Wang. 2016. An ontology-based approach to security pattern selection. *International Journal of Automation and Computing* 13, 2 (2016), 168–182.

[11] Amelie Gyrard, Soumya Kanti Datta, and Christian Bonnet. 2018. A survey and analysis of ontology-based software tools for semantic interoperability in IoT and WoT landscapes. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. IEEE, 86–91.

[12] Atsuo Hazeyama, Hironori Washizaki, Nobukazu Yoshioka, Haruhiko Kaiya, and Takao Okubo. 2016. Literature survey on technologies for developing privacy-aware software. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*. IEEE, 86–91.

[13] Almut Herzog, Nahid Shahmehri, and Claudiu Duma. 2007. An ontology of information security. *International Journal of Information Security and Privacy (IJISP)* 1, 4 (2007), 1–23.

[14] Raja Khaim, Saba Naz, Fakhar Abbas, Naila Iqbal, Memoona Hamayun, and Rawalpindi Pakistan. 2016. A review of security integration technique in agile software development. *Int. J. Softw. Eng. Appl* 7, 3 (2016), 49–68.

[15] A. Koivu, L. Koivunen, S. Hosseinzadeh, S. Laurén, S. Hyrynsalmi, S. Rauti, and V. Leppänen. 2016. Software Security Considerations for IoT. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 392–397. https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.93

[16] Rajeev Kumar, Mohammad Zarour, Mamdouh Alenezi, Alka Agrawal, and Raees Ahmad Khan. 2019. Measuring security durability of software through fuzzy-based decision-making process. *International Journal of Computational Intelligence Systems* 12, 2 (2019), 627–642.

[17] Birger Lantow. 2016. OntoMetrics: Putting Metrics into Use for Ontology Evaluation. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016)*. SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT, 186–191. https://doi.org/10.5220/0006084601860191

[18] Antoni Lluís Mesquida and Antonia Mas. 2015. Implementing information security best practices on software lifecycle processes: The ISO/IEC 15504 Security Extension. *Computers & Security* 48 (2015), 19–34.

[19] Hanif Mohaddes Deylami, Iman Ardekani, Ravie Chandren Muniyandi, and Hossein Sarrafzadeh. 2015. Effects of software security on software development life cycle and related security issues. (2015).

[20] Nabil M Mohammed, Mahmood Niazi, Mohammad Alshayeb, and Sajjad Mahmood. 2017. Exploring software security approaches in software development lifecycle: A systematic mapping study. *Computer Standards & Interfaces* 50 (2017), 107–115.

[21] Azzam Mourad, Mourad Debbabi, and Marc-André Laverdiere. 2008. Security hardening of open source software. (2008).

[22] Natalya F Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W Fergerson, and Mark A Musen. 2001. Creating semantic web contents with protege-2000. *IEEE intelligent systems* 16, 2 (2001), 60–71.

[23] Francisco José Barreto Nunes and Adriano Bessa Albuquerque. 2010. A Secure Software Development Supported by Knowledge Management. In *Innovations and Advances in Computer Sciences and Engineering*, Tarek Sobh (Ed.). Springer Netherlands, Dordrecht, 291–296.

[24] María Poveda-Villalón, Mari Carmen Suárez-Figueroa, and Asunción Gómez-Pérez. 2012. Validating ontologies with oops!. In *International conference on knowledge engineering and knowledge management*. Springer, 267–281.

[25] Nucharee Premchaiswadi. 2018. Security Management and Knowledge Engineering. *Engineering Journal of Siam University* 29 (2018).

[26] Akond Ashfaque Ur Rahman and Laurie Williams. 2016. Software security in devops: Synthesizing practitioners' perceptions and practices. In *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*. IEEE, 70–76.

[27] Muthu Ramachandran. 2016. Software security requirements management as an emerging cloud computing service. *International Journal of Information Management* 36, 4 (2016), 580–590.

[28] Widura Schwittek, Holger Schmidt, Kristian Beckers, Stefan Eicker, Stephan Faßbender, and Maritta Heisel. 2012. A common body of knowledge for engineering secure software and services. In *2012 Seventh International Conference on Availability, Reliability and Security*. IEEE, 499–506.

[29] Romilla Syed and Haonan Zhong. 2018. Cybersecurity Vulnerability Management: An Ontology-Based Conceptual Model. (2018).

[30] Artem Vorobiev, Nargiza Bekmamedova, et al. 2010. An ontology-driven approach applied to information security. *Journal of Research and Practice in Information Technology* 42, 1 (2010), 61.

[31] S. Wen. 2017. Software security in open source development: A systematic literature review. In *2017 21st Conference of Open Innovations Association (FRUCT)*. 364–373. https://doi.org/10.23919/FRUCT.2017.8250205

[32] Shao-Fang Wen and Basel Katt. 2018. An Ontology-Based Context Model for Managing Security Knowledge in Software Development. In *Proceedings of the 23rd Conference of Open Innovations Association FRUCT*. FRUCT Oy, 56.