information

MDPI

Article

Managing Software Security Knowledge in Context: An Ontology Based Approach

Shao-Fang Wen * and Basel Katt

Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, Gjøvik 2815, Norway; basel.katt@ntnu.no

* Correspondence: shao-fang.wen@ntnu.no

Received: 29 May 2019; Accepted: 18 June 2019; Published: 20 June 2019

Abstract: Knowledge of software security is highly complex since it is quite context-specific and can be applied in diverse ways. To secure software development, software developers require not only knowledge about general security concepts but also about the context for which the software is being developed. With traditional security-centric knowledge formats, it is difficult for developers or knowledge users to retrieve their required security information based on the requirements of software products and development technologies. In order to effectively regulate the operation of security knowledge and be an essential part of practical software development practices, we argue that security knowledge must first incorporate features that specify what contextual characteristics are to be handled, and represent the security knowledge in a format that is understandable and acceptable to the individuals. This study introduces a novel ontology approach for modeling security knowledge with a context-based approach, by which security knowledge can be retrieved, taking the context of the software application at hand into consideration. In this paper, we present our security ontology with the design concepts and the corresponding evaluation process.

Keywords: software security; knowledge management; security ontology; context-based

1. Introduction

Knowledge of software security is highly complex since it is quite context-specific and can be applied in diverse ways [1]. Software developers not only require knowledge about general security concepts but also need expertise to deal with the variant technologies, frameworks, and libraries involved with software development projects [2]. The complex security knowledge usually surpasses the capacity of software developers to solve security problems by themselves [3]. For example, the security principle of least privilege recommends that accounts should have the least amount of privilege required to perform the task. This encompasses security practices of user rights, and resource permission such as CPU, memory, and network, which exist with different programming languages (e.g., C, C++, PHP, Java and so on), depending on the features of the software products. However, much of the required security knowledge is traditionally encapsulated in unstructured or semi-structured formats [4] and commonly organized in a security-centric structure, as either black-hat or white-hat security. With such topical security knowledge formats, it is difficult for developers or knowledge users to retrieve the required security information based on the requirements of software products and development technologies.

In order to effectively operate security knowledge and be an indispensable part of practical software development practices, we argue that security knowledge must first incorporate additional contextual features, that is, to contextualize security knowledge with certain characteristics of software applications, and then represent it in a format that is understandable and acceptable to the individuals. Ontology has been regarded as a good knowledge management approach in the domain of information

security to methodically classify various security concepts, such as security attacks and vulnerabilities as well as related security prevention mechanisms [5,6]. The knowledge representation of ontology not only integrates knowledge resources at both abstract and semantic levels, but can also be adopted by knowledge sharing services such as advanced knowledge search, knowledge visualization, and therefore, supports the learning process of software security.

This paper is part of ongoing research developing a contextual learning environment for software security, in which ontology is used as the kernel knowledge repository in managing contextualized security knowledge. The objective of this research is to support software developers and knowledge users to define and use security knowledge appropriately, adapting to their working context. The ontology we designed integrates application context, security domain knowledge, and contextualized knowledge, allowing contextual inquiry through software scenarios that users would be interested in or familiar with. In this paper, we present our security ontology with the design concepts and the evaluation process.

The rest of this paper is organized as follows. Section 2 introduces the background knowledge about the context and knowledge. In Section 3, we describe the design of our ontology. Section 4 presents the evaluation process of the ontology, followed by a discussion in Section 5. We discuss related works in Section 6. Lastly, Section 7 presents the conclusion and our future works.

2. Context and Knowledge Management

According to Brézillon [7], "context is a set of information used to characterize a situation in which human and computational agents interact". He also points out that knowledge comes from a variety of contexts that cannot be accurately understood out of context [8,9]. The context has the ability to provide major meaning to knowledge, promoting more effective comprehension of the determined situation in the collaborative work [10]. Context is a crucial component of fully understanding of knowledge [8,11,12], which promotes more effective comprehension of a determined situation in the collaborative work [10]. Without appropriate contextual description, knowledge could be isolated from other relevant knowledge, resulting in limited or distorted understanding [9,13]. Since context can provide rich information about why and where a piece of knowledge is applied, it is very necessary to consider the context during the use of the knowledge to improve its applicability [14].

Knowledge management is defined as "the capability by which communities capture the knowledge that is critical to their success, constantly improve it, and make it available in the most effective manner to those who need it" [15]. Context has been considered a critical concept in knowledge management, where the relevant architectures should include the design of knowledge elements as well as the design of the overall contextual characteristics of the knowledge and the relationships between them [16]. In this situation, knowledge artifacts need to be equipped with context-based features so that they can distribute information effectively within the application domain and relate more evenly to other specific knowledge across the organization [17].

3. Design of the Ontology

The basic design concept of our ontology was to build linkages with contextual software scenarios (according to the application context). The corresponding contextualized security knowledge from the critical security concepts were drawn from the security domain model as a common vocabulary (see Figure 1).

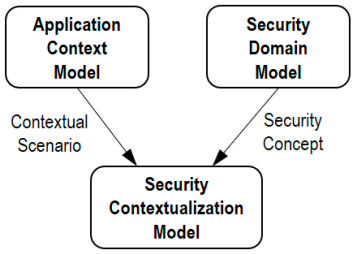

Figure 1. Three models span the modeling of contextualized security knowledge.

3.1. Application Context Modeling

The context model defined a complete representation of what context is in a particular domain. In our ontology, the context for software security knowledge is supported by the creation of scenarios in different application contexts. Contextual scenarios refer to different manifestations within a context [18]. The scenario presents a snapshot of possible features and corresponding code fragments in the specific functionality. For example, regarding the application functionality of "Generating HTML pages" in web application context there includes a set of scenarios, such as generating static or dynamic pages, and using external data from HTTP requests or data stores. We choose a scenario-based approach because scenarios can be easily adapted to the situation of the represented applications and can be easily integrated with the conceptual security knowledge. It also draws on situated security knowledge, that is, understandings particular to the application context in which they generate. Figure 2 represents the application context model used in the ontology. In the context modeling, in addition to scenarios, we focus on characteristics that are highly relevant for retrieval within a software application, concerning three perspectives:

- The functional area (and the corresponding functionalities) that the application is associated with.
- The application category that scenario/functionality belongs to.
- The platforms that the scenario functionality is used.

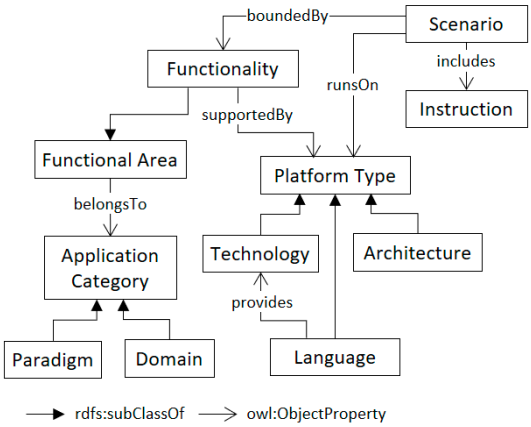

Figure 2. Application Context Model.

- *Application category*: It is a set of characteristics to categorize software applications, in which two sub-classes are included: *Paradigms* (e.g., web, mobile, and desktop applications, etc.) and *Domains* (e.g., banking, health, and logistics applications, etc.).
- *Platform type*: This superclass specifies programming languages, technologies, and architectures that are used to create the software application. *Technology* can be provided by a certain programming language. For example, Silverlight is the technology that has been implemented in C# language, while J2EE is the subset of Java technologies. *Architectures* refer to the fundamental system structure used to operate the application, such as the MySQL database management system and an Android operating system.

- *Functional area*: It is a group of application functionalities, which represents an aspect of software applications that can be performed by users or other systems in a particular application category. For example, "Outputting HTML" is a functional area in web applications paradigm, in which "Generating HTML dynamically using user-supplied data" is one of the functionalities. A functionality is supported and run on some combination of platform types.

3.2. Security Domain Modeling

The security domain model describes knowledge as teaching subjections through a set of concepts. Figure 3 illustrates the security concepts and their relationships in the security domain model. In this model, we aim to define security knowledge schema that easily formed as metal models of learners learning about software security. For this purpose, the schema should be simplified and remain focused on the objective of reducing the content load. In general, our intention is to guide users in answering three questions while dealing with software scenarios: (1) What are the possible attacks? (2) Why does the software encounter attacks? (3) How can these attacks be prevented or mitigated? In accordance with such design considerations, we identified three security concepts that are most widely used throughout the security domain and need to be concentrated learning on. Ultimately, three classes were incorporated into the security conceptualization model: *Security Attack*, *Security Weakness*, and *Security Practice*. The definitions of the three security concepts are given in the following:

- *Security Attack*: This represents actions taken against the software application with the intention of doing harm. Examples are SQL injection, Cross-Site Scripting (XSS), etc. Security attacks exploit security weakness existed in software applications.
- *Security Practice*: This represents methods, procedures or techniques to prevent security weakness. Examples are "Input validation" and "Output encoding" in preventing XSS.
- *Security Weakness*: This represents bug, flaws, vulnerabilities and other errors exist in the software applications. Examples are "Improper to neutralize input during HTML generation" and "Fail to perform a bound check while copying data into memory stack".

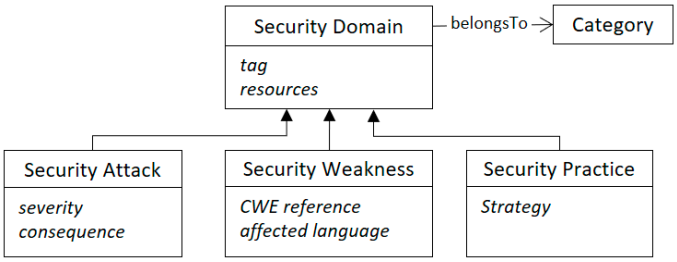

Figure 3. Security domain Model.

From a security conceptualization point of view, we only want to indicate which principles or abstract ideas are needed, not their practical implementation. Therefore, we describe security knowledge in this model at a level of abstraction. The instances of these classes specify only the fundamental characteristics of the security concepts, not specific software application aspects. The major advantage of this design is how it enhances comprehension of the conceptual security knowledge among various security contexts. Furthermore, we adopt an abstract *Security Domain* class as a superclass for all security concepts. In the security conceptualization model, we apply segmentation of interests so that only generic descriptions remain as attributes in the class *Security Domain*. Additionally, we created a *Category* class, in which security concepts were grouped into categories.

3.3. Security Contextualization Modeling

To help users gain a more flexible understanding of the study concept in a range of situations with varying levels of abstraction, we organize security knowledge by blending abstract and concrete

perspectives. The term contextualization is used here to describe the process of drawing specific connections between security domain knowledge being taught and an application context in which the abstract knowledge can be relevantly applied or illustrated. In this study, abstract knowledge refers to the conceptual security domain knowledge, while concrete knowledge relates to the contextualized scenario-specific security knowledge. Research has shown that presenting knowledge in both concrete and abstract terms are far more powerful than presenting either one in isolation [19].

To this extent, the security contextualization modeling manages security knowledge in the context of specific scenarios and brings together the conceptual knowledge that is described in the security conceptualization model. The including security concepts are aligned with those defined in the security conceptualization model, which are *Security Attack*, *Security Weakness*, and *Security Practice*. In order to clearly state the purposes and distinguish them from the security conceptualization model, we used different classes, namely *Concrete Security Attack*, *Concrete Security Weakness*, and *Concrete Security Practice*. Figure 4 illustrates the security contextualization modeling. The abstract class *Contextualized Knowledge* is used from which these three classes inherit common attributes such as tags or external resources. Once the conceptualization knowledge model is defined, each security concept can be connected to the corresponding classes in the security conceptualization model. Figure 5 depicts the full view of the ontology-based knowledge model, including the interrelationships of the components.

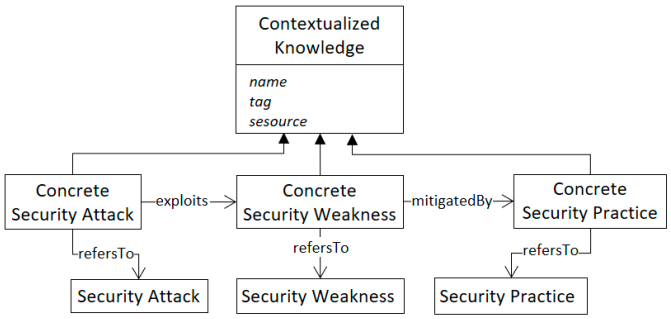

Figure 4. Security contextualization model.

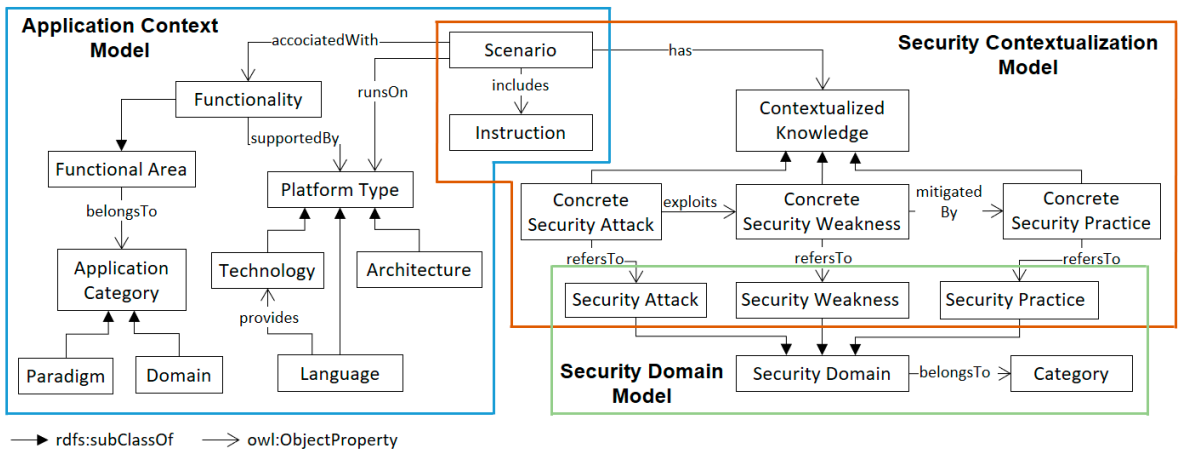

Figure 5. The ontology-based security knowledge model.

4. Evaluation of the Ontology

To validate the effectiveness of the ontology, we conducted several evaluation phases. The overall evaluation process that we undertook is shown in Figure 6.

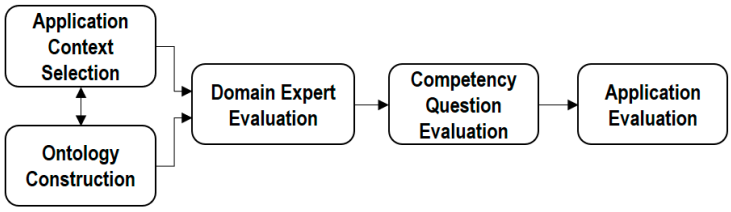

Figure 6. The ontology evaluation process.

First, in order to evaluate the proposed ontology with a real-world case, we chose a Web Application paradigm with Flat PHP technology as the application context of this pilot study. Functionalities, scenarios and security knowledge items (attacks, weakness, and practices) were collected under the defined context. The ontology (concepts and relationships) were implemented using the Protégé tool [20] with Ontology Web Language (OWL, https://www.w3.org/OWL). Figure 7 depicts the ontology design in Protégé editor whereas Figure 8 presents the maintenance of object properties and data properties for contextualized knowledge (Security Attack).

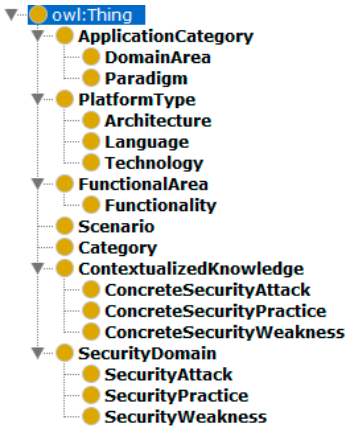

Figure 7. Ontology design in Protégé editor.

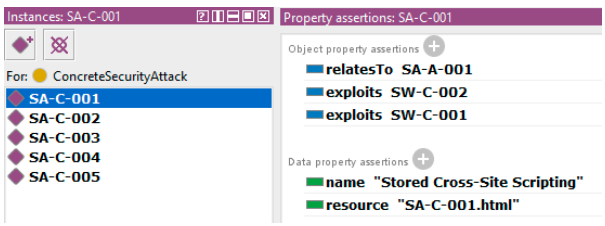

Figure 8. The objective property and data property of concrete knowledge (Security Attack).

The domain expert evaluation was carried out by an internal security professional within NTNU who provided the competencies using a computer/cybersecurity, and ontology building method and analysis. The ontology structure, including concept definitions and relations, were reviewed and analyzed. A few weaknesses were identified:

(1) Difficulty to model software technologies and architectures in application context model,
(2) No category classes to group knowledge items in the security domain model, and
(3) No vulnerability concepts in the security domain model.

We considered comments (1) and (2), and have issued change requests of the ontology design, in which a *Category* class was created in the security domain model, whereas the class of *Platform type* was split into two sub-concepts, namely *Technology* and *Architecture*. In the domain model, we did not differentiate between terms *Security Weakness* and *Vulnerability*, as *Security Weakness* is naturally a

more general class that could cover different security errors, such as design flaw and coding errors. Therefore, the idea in (3), which suggested incorporating concepts of vulnerability, was shelved.

After taking the review comments and mitigating the identified weakness, the ontology was evaluated with competency questions against its initial requirements. Therefore, two exemplary questions were developed:

Q1: What are the available software scenarios given in the functionality "Generating output in web pages using user-supplied data", PHP language and MySQL database?

Q2: What are the relevant contextualized knowledge items of the first scenario from the result of the question (a)?

To answer the above competency questions, we used SPARQL protocol [21] to extract information from the RDF graph. Two corresponding SPARQL statements were prepared and executed in Protégé editor. Figure 9 demonstrates querying scenarios using the given functionality and platform types (programming language and architecture), from which *Q1* can be answered. For the answer of *Q2*, Figure 10 shows the query result that returns the instances of contextualized security knowledge of a specific scenario, and the short names of related security domain knowledge.

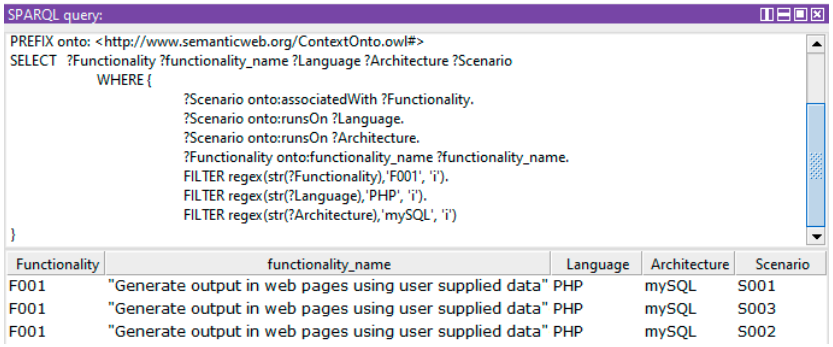

Figure 9. An example of SPARQL (to query Scenarios).

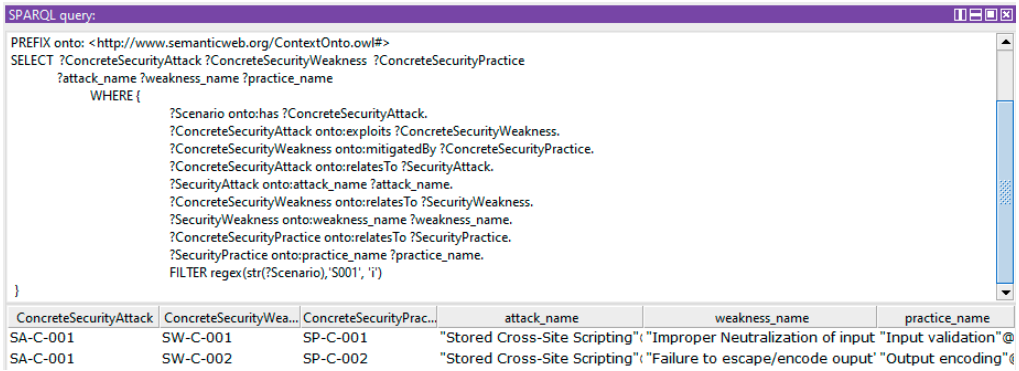

Figure 10. An example of SPARQL (to query security knowledge).

After the domain expert's review with a competency-question examination, we took a further application-based evaluation approach [22,23] by plugging the ontology into an application for further evaluation. For this purpose, we developed a web-based application prototype based on our proposed ontology. The objective of this application is to present scenario-based security knowledge that is both concrete and abstract, according to the contextual information that the user provides. The front-end was designed as a web-based user interface with HTML and JavaScript languages. The back-end was implemented with Java, and the ontology repository was accessed with Jena API (https://jena.apache.org), which is a Java framework used for building semantic web applications. Jena has the advantage that it provides wild-ranging Java libraries to help developers handle OWL, and SPARQL conforming to W3C recommendations. Figure 11 presents the user interface of the

context menu, in which the learner selects relevant criteria based on the desired knowledge (or prior programming experience) to scope the functionalities and corresponding scenarios.

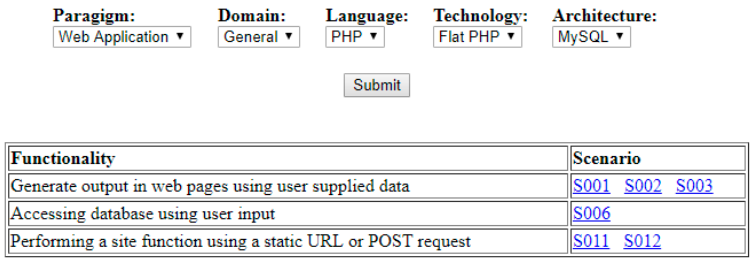

Figure 11. The user interfaces for context selection.

In presenting the security knowledge, the web page is mainly composed of four framesets: the security knowledge structure, the scenario description, the contextualized knowledge, and conceptual knowledge. Figure 12 shows a snapshot of the knowledge presentation. The scenario here is used as a starting point to browse security knowledge, which is made up of practical demonstrations of the pre-described application functionality and the code fragments that were extracted from the class Scenario and Instruction. To present practical security knowledge for the scenario, that is contextualized knowledge, we extracted information from classes under the Contextualized Knowledge superclass, which includes perceptually detailed and rich materials from ontology, such as security attacks with different exploits, coding mistakes, and the corresponding secure coding practices. In addition to the practical knowledge, users can also capture abstract explanation as well, that is conceptual knowledge from Security Domain classes.

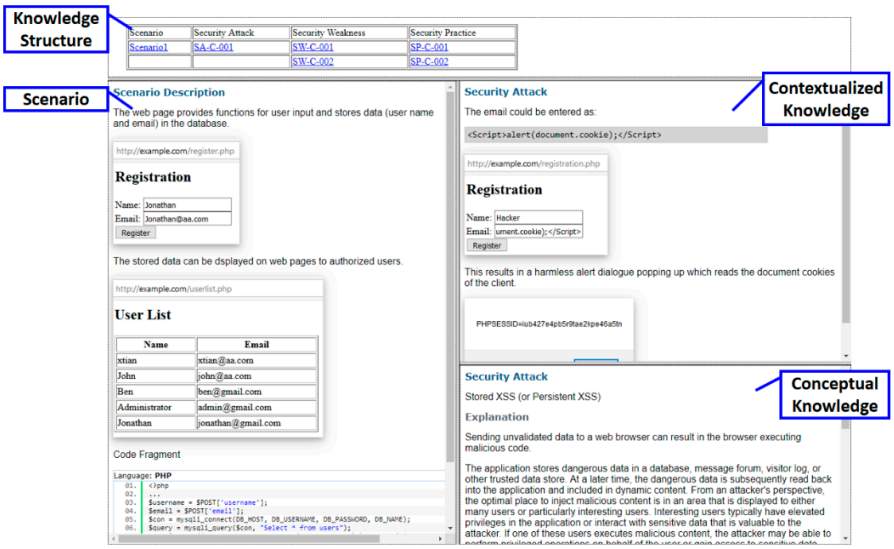

Figure 12. The user interfaces for security knowledge presentation.

5. Discussion

Ontology technologies have become the core component of today's applications such as electronic commerce, knowledge portals, information integration and sharing, and web services [24–29]. Our ontology approached the role of ontologies in managing contextualized knowledge in the domain of software security. With the context-based design approach, a dynamic situational application scenario can be integrated together with the conceptual security domain knowledge. The advantage of this ontology model is two-fold. First, it separates concrete and abstract security knowledge in two models, which simplifies knowledge maintenance and retrieval. Second, it shares a common

understanding of security concepts between security domain and contextualization models to enable semantic interoperability.

In addition to the application scenario demonstrated in the previous section, this ontology can be also used in various settings. For example, in the pedagogical environment, a course tutor, who is engaged in the introduction of security vulnerabilities, can use the proposed ontology to quickly identify a number of real-world examples of facing a specific security attack or vulnerability, to improve the effectiveness of learning. In the practical software development process, software engineers are allowed to find solutions to exceptional situations by searching for similar contexts. For example, a PHP web application designer can refer to another security setup by looking for a similar domain and software technologies. The presence of detailed information on the relation between classes can enable answering the various questions related to security tasks. Furthermore, since our ontology is developed using the OWL standard in the Protégé tool, it enables the possibility to be used by an automated tool to provide advanced services such as more accurate security requirements and design suggestions.

Yet, the software security domain is complex and dynamic. New threats and countermeasures are continuously evolving. Although the approach described here provides technologies to store and present security knowledge, security experts or practitioners' involvement is crucial to fill the security ontologies with the necessary information and then to apply them in security education and the practical software development process.

6. Related Work

There has been extensive research in the area of security knowledge modeling and ontology applications to software security. Some papers focus on using an ontology to model security vulnerabilities. Guo and Wang [30] presented an ontology-based approach to model security vulnerabilities listed in CVE (Common Vulnerability and Exposure, https://cve.mitre.org/). The authors identified critical security vulnerabilities in the domain of software security, which can provide machine-understandable CVE vulnerability knowledge and reusable security vulnerability interoperability. Syed and Zhong [31] proposed an ontology-based conceptual model for the formal knowledge representation of the cybersecurity vulnerability domain and intelligence, which integrated cybersecurity vulnerability concepts from several sources including CVE, NVD, CVSS (Common Vulnerability Scoring System, https://www.first.org/cvss/) framework, and social media. Alqahtani et al. [32] proposed an ontological representation, which establishes links with bi-directional traceability between traditional software repositories (e.g., issue trackers, version control systems, Q and A repositories) and security vulnerabilities databases (e.g., NVD (National Vulnerability Database, https://nvd.nist.gov/))

Some researchers presented their ontology in support of security requirements and design processes in software development. Gyrard et al. [33] proposed STACK ontology (Security Toolbox: Attacks and Countermeasures) that supported developers in the design of secure applications. Countermeasures in STACK included cryptographic concepts (encryption algorithms, key management, digital signature, and hash function), security tools, and security protocols. Kang and Liang [34] presented the security ontology adopting model driven architecture (MDA) methodology. Their proposed ontology could be used in security concept modeling in each phase of the development process (e.g., the requirement and design phases) with MDA. In order to improve the application of security patterns of the security engineering domain, Guan et al. [35] proposed an ontological approach facilitating security knowledge mapping from security requirements to security patterns. Manzoor et al. [36] developed an ontology, illustrating the relationships across various actors involved in the Cloud ecosystem, to analyze different threats to/from Cloud-system actors.

Finally, some efforts focused on building security ontology specifically in the context of web application development. Salini and Kanmani [37] presented an ontology for defining the security requirements of web applications. The included concepts are assets, vulnerabilities, threats, and

stakeholders. Their ontology aimed at reusing the knowledge of security requirements in the development of different kinds of web applications. Buch and Wirsing [38] presented a security ontology for secure web applications (SecWAO), which aimed to support web developers to specify security requirements or make design decisions in web application development. It distinguished various concepts among methods, tools, mechanism, assets, vulnerabilities, and threats. Velasco et al. [39] presented an ontology-based framework for string, presenting and reusing security requirements. Their framework integrated security standards, methods of risk analysis, and ontology requirements.

A major feature, which is common for all of the above studies, is that the ontology is security driven, focusing on unifying security concepts and terminology. Subsequently, they are either dedicates to a certain software domain or support part(s) of the software development process. Our ontology approach differentiates from the previous research work in the following aspects:

(1) Our ontology is context-based, which models security knowledge with a diversity of software features and technologies;

(2) Our ontology describes security knowledge with a contextual situation and, meanwhile, complements the concrete knowledge with abstract description.

7. Conclusions and Future Work

This paper presents a novel approach for modeling software-security knowledge with a context-based approach, in which the security knowledge can be retrieved taking the context of the software application into consideration. The design of our ontology ensures that users understand the security-relevant aspects of critical software features. In addition, software developers are able to efficiently identify the possible attacks and security errors associated with the functionalities of their software products based on the domain of the application, the programming language or yjr technologies. In this paper we presented the core concepts of the ontology, as well as an evaluation with an application scenario. Our proposal was deemed useful for security researchers who wish to formalize and manage contextualized knowledge in their domain, systems, and methods.

In future work we expect to expand the ontology continuously, enriching the knowledge content by including more software scenarios with a broad application context while also providing contextual details in the branches of security domain knowledge and enriching the abstract explanations. We also plan to further evaluate the modeling approach with educators and security experts in the domain of information security. We believe that such a context-based approach in ontology modeling can benefit border security domains, such as network security and cryptography. Furthermore, we intend to enhance and complete a learning system for software security based on this ontology. The ultimate goal of our research is to create conditions for more effective learning about software security, which can motivate learners and stimulate their interest.

Author Contributions: Initial conceptualization and framework were developed by S.-F.W. The methodology and the core part of the ontology were reviewed by B.K. Manuscript was largely written by S.-F.W. Final paper review and editing were performed by B.K.

Funding: This research received no external funding.

Conflicts of Interest: The author declares that they have no conflict of interest.

References

1. McGraw, G. *Software Security: Building Security In*; Addison-Wesley Professional: Boston, MA, USA, 2006; Volume 1.
2. Rus, I.; Lindvall, M. Knowledge management in software engineering. *IEEE Softw.* **2002**, *19*, 26. [CrossRef]
3. Henninger, S. Case-based knowledge management tools for software development. *Automated Softw. Eng.* **1997**, *4*, 319–340. [CrossRef]
4. Cheng, C.K.; Kurfess, F.J. A Context-Based Knowledge Management Framework for Software Development. *Int. J. Comp. Integr. Man.* **2009**, *22*, 1073–1088.

5. Fenz, S.; Ekelhart, A. Formalizing information security knowledge. In Proceedings of the 4th international Symposium on information, Computer, and Communications Security, Sydney, Australia, 10–12 March 2009; pp. 183–194.
6. Tsoumas, B.; Gritzalis, D. Towards an ontology-based security management. In Proceedings of the 20th International Conference on Advanced Information Networking and Applications, Vienna, Austria, 18–20 April 2006; pp. 985–992.
7. Brézillon, P. Making context explicit in communicating objects. In *Communicating with Smart Objects: Developing Technology for Usable Pervasive Computing Systems*; ISTE Publishing Company: London, UK, 2003.
8. Brézillon, P. Modeling and Using Context: Past, Present and Future. Available online: http://ftp.lip6.fr/lip6/reports/2002/lip6.2002.010.pdf (accessed on 23 March 2019).
9. Brézillon, P.; Pomerol, J.-C. Contextual knowledge sharing and cooperation in intelligent assistant systems. *Le Travail Humain* **1999**, *62*, 223–246.
10. Brézillon, P.; Araujo, R. Reinforcing shared context to improve collaboration. *Revue d'Intel. Artif.* **2005**, *19*, 537–556. [CrossRef]
11. Klemke, R. Context Framework - an Open Approach to Enhance Organisational Memory Systems with Context Modelling Techniques. In Proceedings of the Third International Conference on Practical Aspects of Knowledge Management (PAKM2000), Basel, Switzerland, 30–31 October 2000.
12. Jafari, M.; Fathian, M.; Jahani, A.; Akhavan, P. Exploring the contextual dimensions of organization from knowledge management perspective. *VINE* **2008**, *38*, 53–71. [CrossRef]
13. Goldkuhl, G.; Braf, E. Contextual knowledge analysis-understanding knowledge and its relations to action and communication. In Proceedings of the Second European Conference on Knowledge Management, Bled, Slovenia, 8–9 November 2001; pp. 197–208.
14. Bishop, M. A Clinic for" Secure" Programming. *IEEE Secur. Priv.* **2010**, *8*. [CrossRef]
15. Birkenkrahe, M. How large multi-nationals manage their knowledge. *Bus. Rev.* **2002**, *4*, 2–12.
16. Rosa, M.G.; Borges, M.R.; Santoro, F.M. A conceptual framework for analyzing the use of context in groupware. In *Groupware: Design, Implementation, and Use*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 300–313.
17. Curtis, B.; Krasner, H.; Iscoe, N. A field study of the software design process for large systems. *Comm. ACM* **1988**, *31*, 1268–1287. [CrossRef]
18. Errington, E.P. Being there: Closing the gap between learners sand contextual knowledge using near-world scenarios. *Int. J. Learn.* **2009**, *16*, 585–594.
19. Pashler, H.; Bain, P.M.; Bottge, B.A.; Graesser, A.; Koedinger, K.; McDaniel, M.; Metcalfe, J. *Organizing Instruction and Study to Improve Student Learning*; U.S. Department of Education, National Center for Education Research: Washington, DC, USA. Available online: https://files.eric.ed.gov/fulltext/ED498555.pdf (accessed on 23 March 2019).
20. Tudorache, T.; Nyulas, C.; Noy, N.F.; Musen, M.A. WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semant. Web* **2013**, *4*, 89–99.
21. Harris, S.; Seaborne, A.; Prud'hommeaux, E. SPARQL 1.1 query language. 2013. Available online: https://www.w3.org/TR/sparql11-query/ (accessed on 23 March 2019).
22. Brank, J.; Grobelnik, M.; Mladenic, D. A survey of ontology evaluation techniques. In Proceedings of the conference on data mining and data warehouses (SiKDD 2005), Ljubljana, Slovenia, 5 October 2015; pp. 166–170.
23. Hlomani, H.; Stacey, D.J. Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey. *Semant. Web Inf. Syst.* **2014**, *1*, 1–11.
24. Gruber, T.R. A translation approach to portable ontology specifications. *Knowl. Acquisit.* **1993**, *5*, 199–220. [CrossRef]
25. Uschold, M.; Gruninger, M. Ontologies: Principles, methods and applications. *Knowl. Eng. Rev.* **1996**, *11*, 93–136. [CrossRef]
26. Noy, N.F.; McGuinness, D.L. *Ontology Development 101: A Guide to Creating Your First Ontology*; Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880; Stanford University: Stanford, CA, USA, 2001.
27. Wang, X.; Dong, J.S.; Chin, C.-Y.; Hettiarachchi, S.; Zhang, D. Semantic space: An infrastructure for smart spaces. *IEEE Perv. Comp.* **2004**, *3*, 32–39. [CrossRef]

28. Gruninger, M. Ontology: Applications and design. *Commun. ACM* **2002**, *45*, 39–41.
29. Patel, C.; Supekar, K.; Lee, Y. OntoGenie: Extracting ontology instances from WWW. In *Human Language Technology for the Semantic Web and Web Services, Proceedings of the 7th IEEE International Symposium on Wearable Computers ISWC'03, White Plains, NY, USA, 21–23 October 2003*; IEEE: Piscataway, NJ, USA.
30. Guo, M.; Wang, J.A. An ontology-based approach to model common vulnerabilities and exposures in information security. In Proceedings of the ASEE 2009 Southest Section Conference, Marietta, GA, USA, 5–7 April 2009.
31. Syed, R.; Zhong, H. Cybersecurity Vulnerability Management: An Ontology-Based Conceptual Model. In Proceedings of the Twenty-fourth Americas Conference on Information Systems, New Orleans, LA, USA, 16–18 August 2018.
32. Alqahtani, S.S.; Eghan, E.E.; Rilling, J. Tracing known security vulnerabilities in software repositories–A Semantic Web enabled modeling approach. *Sci. Comp. Prog.* **2016**, *121*, 153–175. [CrossRef]
33. Gyrard, A.; Bonnet, C.; Boudaoud, K. The stac (security toolbox: Attacks & countermeasures) ontology. In Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 165–166.
34. Kang, W.; Liang, Y. A security ontology with MDA for software development. In Proceedings of the 2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Nanjing, China, 26–29 May 2013; pp. 67–74.
35. Guan, H.; Yang, H.; Wang, J. An ontology-based approach to security pattern selection. *Int. J. Automat. Comp.* **2016**, *13*, 168–182. [CrossRef]
36. Manzoor, S.; Vateva-Gurova, T.; Trapero, R.; Suri, N. Threat Modeling the Cloud: An Ontology Based Approach. In Proceedings of the International Workshop on Information and Operational Technology Security Systems, Crete, Greece, 13 September 2018; pp. 61–72.
37. Salini, P.; Kanmani, S. Ontology-based representation of reusable security requirements for developing secure web applications. *Int. J. Intern. Tech. Secur. Trans.* **2013**, *5*, 63–83. [CrossRef]
38. Busch, M.; Wirsing, M. An Ontology for Secure Web Applications. *Int. J. Softw. Inf.* **2015**, *9*, 233–258.
39. Lasheras, J.; Valencia-García, R.; Tomás Fernández-Breis, J.; Ambrosio Toval Álvarez, J. Modelling reusable security requirements based on an ontology framework. *J. Res. Pract. Inf. Tech.* **2009**, *41*, 119.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).