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: In the setting of software development, knowledge can be both dynamic and situation specific, and the complexity of knowledge usually exceeds the capacity of individuals to solve problems by themselves. Software developers not only require knowledge about the general security concepts but also about the context for which software is being developed. With traditional security knowledge formats, which are usually organized in a security-centric way, it is difficult for knowledge users to retrieve the desired security information to fulfill the requirements of their working context. 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 additional features, that is, to first specify which contextual information is to be handled, and then 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 in a context-sensitive manner where the security knowledge can be retrieved while taking the context of the application in hand into consideration. In this paper, we present our security ontology with the design concepts and the evaluation process.

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

1. Introduction

The domain of software security knowledge is highly complex since it is quite context-specific and can be applied in diverse ways [1]. Software engineers not only require knowledge about the general security concepts but also need the expertise to cope with the variety of frameworks, technologies, and libraries that are involved with software projects [2]. The complexity of security knowledge usually exceeds the capacity of individuals 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 the security practices of user rights, and resource permission such as CPU, memory, and network, which exist for specific programming languages (e.g., C, C++, PHP, Java and so on), depending on the features of the software product. However, much of the required security knowledge is traditionally encapsulated in unstructured or semi-structured formats [4], and commonly organized in a security-centric way, as either back-hat or white-hat security. With such topical security knowledge formats, it is difficult for knowledge users to retrieve the desired security information based on the requirements of software projects or related to their working environments.

Therefore, 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 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. For security knowledge management, Ontology has been regarded as a good approach to systematically classifying and categorizing various security concepts such as attacks and prevention as well as related security implementation technology [5,6]. The ontological representation not only supports the integration of knowledge resources at various abstraction and semantic levels, but it can also be used by knowledge sharing services such as knowledge integration and interoperability, advanced knowledge search, knowledge visualization, and therefore support the sharing and learning process about software security.

This paper is part of ongoing research of developing a contextual learning environment for software security, in which an ontology is used as the kernel knowledge repository in managing contextualized security knowledge. Our research objective is to support software developers and knowledge users to capture 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 context and it cannot be accurately understood without context [8,9]. The context has the capacity to provide a major meaning to knowledge, promoting a more effective comprehension about a determined situation in the collaborative work [10]. Contextual information is a crucial component of fully understanding knowledge [8,11,12]. Without proper contextual information, knowledge can be isolated from other relevant knowledge, resulting in limited or distorted understanding [9,13]. Since context can give guidance about when, where and why a piece of knowledge is used, considering the context in knowledge use is very necessary to enhance the applicability of knowledge [14].

Knowledge management has been 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]. In knowledge management, the context has been considered as a relevant concept. Any architecture of knowledge management should include the design of knowledge items as well as the design of the overall contextual elements of the knowledge and the relationships among them [16]. In this situation, knowledge tools should be equipped with context-carrying functions so that it can effectively disseminate information to spread the application domain and other specific knowledge more evenly across the organization [17].

3. Design of the Ontology

The basic concept of our ontology design is to provide a vocabulary for representing security concepts in the domain of software security, and for providing linkages with contextual scenarios in the application context to formalize contextualized security knowledge (see Figure 1).
3.1. Application Context Modeling

The context model represents a definition of what context is in a specific 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.

- **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 combinations of platform types.

3.2. Security Domain Modeling

The security domain model describes the knowledge that is an object of teaching through a set of concepts (topics to be taught). Figure 3 illustrates the security concepts and their relationships in the security domain model. In this model, we aim to provide security knowledge structure (schema) that is easier to store in the learners’ memory for learning. 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 Domain Model](image)

**Figure 3. Security domain Model.**

- **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”.

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 main advantage of this design is to share a common understanding of the conceptual security knowledge among different security contexts. Furthermore, we adopt an abstract class **Security Domain** as a superclass for all security concepts. In the security conceptualization model, we apply separation of concerns so that only very general descriptions remain as attributes in the class **Security Domain**. Additionally, for convenience, we allow grouping domain knowledge in categories, which themselves can belong to security concepts.

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 our 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 shows the completed 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.
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. We then implemented all the designed classes and corresponding relationships using OWL (https://www.w3.org/OWL/) (Web Ontology Language), a markup language based on RDF/XML (https://www.w3.org/RDF/) (Resource Description Framework/Extensible Markup Language), and we used the Protégé OWL tool to create and maintain the ontology because of its simplicity and popularity [20]. 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).

The domain expert evaluation was carried out by an internal security professional within NTNU who provided the competences using a computer/cybersecurity, and ontology building method and analysis. The ontology structure, including concept definitions, relations, and formal axioms, was 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 elimination of 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.
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 and evaluating the results of the application. For this purpose, we developed a prototype of a web-based application 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 backend was implemented in Java and access to the ontology repository was provided through the Jena API (https://jena.apache.org/), a Java framework for building semantic web applications. Jena provides extensive Java libraries for helping developers develop code that handles RDF, OWL, and SPARQL in line with published 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.

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.

5. Discussion

Ontologies are now central to many applications such as scientific knowledge portals, information management, and integration systems, electronic commerce and web services. The main areas in which ontological modeling is applied, include communication and knowledge sharing, logic inference and reasoning, and knowledge base [24–28]. Given that ontologies are the representation
of the domain knowledge, they explicitly formalize and specify the concepts and their corresponding relationships. Also, they include associated specific instances for the corresponding concepts. These instances contain the actual data used in knowledge-based applications [29]. Specifically, 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 main advantage of this ontology model is to share a common understanding of general security concepts among different software development context to enable semantic interoperability. It also enables the reuse of domain knowledge, i.e. building a common security knowledge base integrated with contextual information, which describes portions of the domain knowledge.

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 have been a number of papers published in the area of security knowledge modeling and applying ontology technologies 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 captured important concepts for describing vulnerabilities in the context of software security, providing machine-understandable CVE vulnerability knowledge and reusable security vulnerabilities interoperability. Syed and Zhong [31] proposed an ontology-based conceptual model for the formal knowledge representation of the cybersecurity vulnerability domain and intelligence, which integrates 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] used a unified ontological representation to establish bi-directional traceability links between security vulnerabilities databases (e.g., NVD (National Vulnerability Database, https://nvd.nist.gov/)) and traditional software repositories (e.g., issue trackers, version control systems, Q&A repositories).

Some research works to present their ontology to support security requirements and design in software development. Gyrard et al. [33] present the STACK ontology (Security Toolbox: Attacks & Countermeasures) to aid developers in the design of secure applications. STACK defines security concepts such as attacks, countermeasures, security properties, and their relationships. Countermeasures can be cryptographic concepts (encryption algorithm, key management, digital signature, and hash function), security tools, or security protocols. Kang and Liang [34] present a security ontology with the Model Driven Architecture (MDA) approach for the use in the software development process. Their proposed security ontology can be used in modeling security concepts in each phase of the development process with MDA. Guan et al. [35] proposed an ontological
approach to facilitates security knowledge mapping from security requirements to security patterns, in order to improve the application of security patterns to security engineering domain. Manzoor et al. [36] developed an ontology, depicting the relationships across the different actors involved in the Cloud ecosystem, to analyze threats to/from Cloud actors.

Finally, some efforts focused on building security ontology specifically in the context of web application development. Salini and Kanmani [37] presented an ontology of security requirements for web applications, including concepts of asset, vulnerabilities, threats, and stakeholders. Their work aims at enabling the reuse of knowledge about security requirements in the development of different web applications. Buch and Wirsing [38] presented the SecWAO ontology with a focus on a secure web application, which aims to support web developers when specifying security requirements or making design decisions. It distinguishes concepts (classes) between methods, notations, tools, categories, assets, security properties, vulnerabilities, and threats. Velasco et al. [39] proposed an ontology-based framework for representing, storing and reusing security requirements. Their framework combines a risk analysis ontology, based on methods of risk analysis and security standards, and a requirements ontology.

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

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

2. Our ontology manages security knowledge in a coherent way of describing it with a contextual situation and incorporating theoretical knowledge to complement the concrete description.

7. Conclusion and Future Work

This paper introduces a novel approach for modeling security knowledge in a context-sensitive manner where the security knowledge can be retrieved while taking the context of the application in hand into consideration. Ontologies make it possible to provide this kind of purpose since it facilitates capture and construction of domain knowledge and enables the representation of skeletal knowledge to facilitate the integration of knowledge bases irrespective of the heterogeneity of knowledge sources [40]. The design of our ontology ensures that users understand the security-relevant aspects of critical software features. In addition, software development staff can be primed with the possible attacks and security errors associated with the software features of their products based on the domain of the application, the programming language they used, and technologies.

In this paper, we have presented the core concepts of the ontology, as well as an evaluation with an application scenario. Our proposal is meant to be useful for security researchers who wish to formalize contextualized knowledge in their domain, systems, and methods.

In future work, we expect to continue expanding the ontology and increasing its expressivity by including more software scenarios with a broad application context, while also providing contextual details in branches of security concepts and enriching the abstract explanations. We also expect to validate the modeling concepts with the collaboration of educators in the domain of information security. We believe that such a context-based ontology approach can benefit border research domains, such as network security and cryptography. Furthermore, we intend to enhance and complete the prototyped 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 Shao-Fang Wen. The methodology and the core part of the ontology were reviewed by Basel Katt. Manuscript was largely written by Shao-Fang Wen. Final paper review and editing were performed by Basel Katt.

Funding: This research received no external funding.
Conflicts of Interest: The author declares that they have no conflict of interest.

References


