Security Knowledge Management in Open Source Software Communities

Shao-Fang Wen, Mazaher Kianpour and Basel Katt

Norwegian University of Science and Technology, 2815 Gjøvik, Norway
{shao-fang.wen, mazaher.kianpour, basel.katt}@ntnu.no

Abstract. Open source software (OSS) communities are groups of individuals, technical or non-technical, interacting with collaborating peers in online communities of practices to develop OSS, solve particular software problems and exchange ideas. People join OSS communities with a different level of programming skills and experience and might lack formal, college-level software security training. There remains a lot of confusion in participants' mind as to what is secured code and what the project wants. Another problem is that the huge amount of available software security information nowadays has resulted in a form of information overload to software engineers, who usually finish studying it with no clue about how to apply those principles properly to their own applications. This leads to a knowledge gap between knowledge available and knowledge required to build secure applications in the context of software projects. Given the increased importance and complexity of OSS in today's world, lacking proper security knowledge to handle vulnerabilities in OSS development will result in breaches that are more serious in the future. The goal of this research work is to fill the knowledge gap by providing an artifact that would facilitate the effective security-knowledge transferring and learning in the context of OSS development. In this work-in-progress paper, we present our ongoing research work following design science research methodology on the domain problem identification and the development of the artifact.

Keywords: Software security; open source software; knowledge management

1 Introduction

Open source software (OSS) is based on the principle that software programs should be shared freely among users, giving them the possibility of introducing implementations and modifications [1]. To develop OSS, solve particular software problems and exchange ideas, numbers of technical and non-technical individuals interact with collaborating peers in online communities of practices [2-4]. The activities that these communities perform are usually called OSS projects. Because of its low-cost software solutions, and the openness and real collaboration of the software development process, OSS has become increasingly popular choice instead of closed source (proprietary) software: about 80% of companies run their operations on OSS [5] and 96% of applications utilize OSS as the software components [6].

As OSS becomes an increasingly important part of our lives, researchers and security communities have spent numerous efforts on providing mechanisms of building security in OSS development [7]. However, the number of new vulnerabilities keeps increasing in today's OSS applications. The Blackduck 2017 Open Source Security and Risk Analysis report announced that 3623 new OSS vulnerabilities occurred in 2016 – almost 10 per day on average and a 10% increase from 2015 [6]. These known vulnerabilities open some of the most critical OSS projects to potential exploitation such as Heartbleed and Logjam (in OpenSSL); Quadrooter (in Android); Glibc Vulnerability (in Linux servers and web frameworks); NetUSB (in Linux kernel), and many others [8, 9]. The increasing quantity and severity of vulnerabilities in OSS have exposed the unique management and software security challenges in OSS communities.

People join OSS communities at different ages and have different backgrounds, capacities, and resources, as well as different objectives. They come from many disciplines and have different levels of programming skills and experience [10], and might lack formal, college-level software security training. When performing contribution, most project participants primarily focus on their immediate goals that usually involve functional requirements and performance [11], and then patch whatever bugs there may be when it's time for the next release or hotfix [12]. There remains a lot of confusion in participants' mind as to what is secured code and what the project wants. Another problem is that the domain knowledge of software security is quite vast and extensive [13]. The huge amount of software security information has resulted in a form of information overload to software engineers who usually finish studying it with no clue about how to apply those principles to their own applications, or with the feeling that security is so difficult to achieve, that they simply cast it aside [14]. Even be educated with software security practices before joining OSS projects these participants may fail to correlate the security knowledge with their projects and fix vulnerabilities in the code when given a chance. It especially relates to what is known as a knowledge gap between knowledge available and knowledge required to build secure applications in the context of software projects [15].

Given the increased importance and complexity of OSS in today's world, lacking proper security knowledge to handle vulnerabilities in OSS development will result in breaches that are more serious in the future. Our position is that there is a need by forging a solution to adaptively place the security knowledge in the appropriate context of the OSS development, i.e., to transfer the necessary security knowledge within the OSS community, meanwhile, to offer opportunities for learning security knowledge and skills to secure OSS products, which are developed, delivered and maintained by the community. The goal of this research work is to propose an artifact that would facilitate the effective security-knowledge transferring and learning in the context of OSS development. In this work-in-progress paper, we present our ongoing work on the domain problem investigation and the conceptual design of the artifact.

The rest of the paper is structured as follows: In section 2, we present an overview of security knowledge management in software development. The research design is presented in section 3. In section 4, we present our initial study of security knowledge sharing and learning in OSS communities. Our proposed knowledge management application and the corresponding research steps are described in section 5 and section 6. Conclusions and future works are presented in section 7.

2 Security Knowledge Management in the Software Development

Software security is more than just a security feature. Security features, such as password encryption and SSL (Secure Socket Layer) between the web server and a browser, are functions of an application to prevent malicious attacks. Security is an emergent, system-wide property of a software system, which means that one cannot presume to achieve a high level of security by simply introducing security-related features into the software [13, 16]. This is because most security problems arise from bugs during the development process [17-19]. Software security aims to avoid security errors in software by considering security aspects throughout the software development lifecycle (SDLC). To train software engineers on critical software security issues, security knowledge should be spread in an effective manner.

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” [20]. Managing knowledge in software development is crucial to allow developers capturing, locating and sharing the knowledge of codes and methods throughout the project to leverage the competitive advantage. In order to increase development staffs’ security knowledge, software project management needs to employ knowledge management mechanisms in encapsulating and spreading the emerging security discipline more efficiently in the software development process. As the software lifecycle unfolds, security-related knowledge could be directly applied with a knowledge-intensive best practice that can support software engineers prevent, spot and mitigate these security errors.

Despite this, our systematic literature review work [7] revealed that no research has been conducted focusing on the aspects of security knowledge management in OSS development. Studies in the areas of software construction and verification (Secure Architecture, Code Review, and Security Testing) are followed by researchers with more interests than governance, where education and training are the major activities. Secure architecture, code review, and security testing can help secure OSS products. However, due to the lack of research and other activities related to security knowledge management for OSS development, software security knowledge cannot be effectively spread within open source communities. Our research intends to fill this research gap by a) empirically investigating the current situation and problems in the knowledge management of software security, b) proposing and developing an application for adaptive security knowledge transferring and learning in OSS communities.

3 Research Design

In developing the security knowledge sharing and learning system adaptive in the context of OSS development, we applied the general methodology of Design Science Research (DSR) framework [21] (Fig. 1) to show how our research is both relevant and rigorous and contribute to the information system (IS) knowledge base by solving a rising security problem in OSS. DSR can be conducted when creating innovations and ideas that define technical capabilities and products through which the development process of artifacts can be accomplished effectively and efficiently [21, 22].

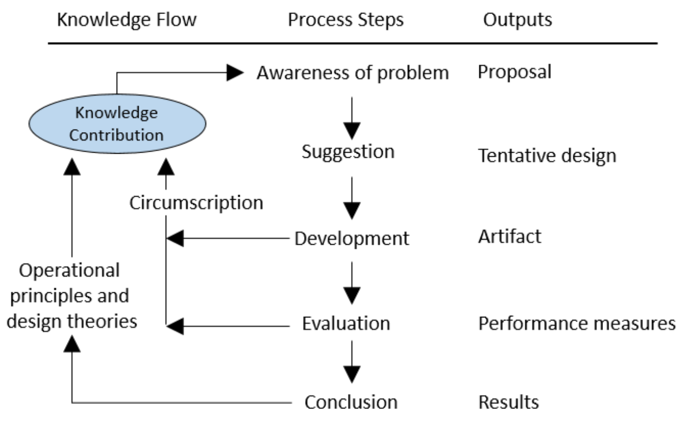


Fig. 1. Design research methodology [21]

4 A Study of the Research Domain Problem

At first, it is important to identify and establish the magnitude of the real-world problem, including technical and non-technical practices in our interests. In contrast to earlier researchers, which have focused on generic learning in OSS communities, our study aimed to observe OSS participants' perception of learning about software security knowledge. Our motivation for this study is not only to evaluate the knowledge sharing and learning mechanisms and the extent to which they may be viable and successful but also to gain insight into the security culture and project factors that affect software-security learning processes in OSS communities.

4.1 Study Method

To get an in-depth understanding of security knowledge sharing and learning behaviors surrounding everyday OSS development, an ethnographic approach was adopted. The ethnographic research can be classified as a qualitative research method that aims to study the cultural patterns and perspectives of participants in their natural settings

[23, 24]. In empirical software engineering, ethnography provides an in-depth understanding of the socio-technical realities surrounding everyday software development practice [23] and highlights the significance of socio-technical issues in the design of software-intensive systems [24]. The knowledge gained can be used to improve processes, methods, and tools as well as to advance the observed security practices

4.1.1 Case Selection.

To get a broader understanding of the phenomena of interests, we set up the following criteria for the case selection: 1) the selected projects should be community driven; 2) the selected projects should be as diverse as possible; 3) the projects use a wide range of communication tools within the communities. Table 1 gives an overview of the selected OSS projects. Having the selected sample cases that cover the range of the diversity of OSS communities is important to refine the phenomena being studied and improve the outcomes of this research endeavor.

Table 1. Overview of the selected projects

Project	Age	Software Category	Programming Language
A	3	Collaborative Text Editor	JavaScript
B	8	Content Management System	PHP
C	5	Multimedia playback engine	C/C++

4.2 Data Collection

Since programming errors have been a major part of security vulnerabilities in software development [25], we dedicated our investigation in the knowledge sharing and learning about secure programming domain knowledge. Two data collection schemes were applied: observation and a semi-structured interview.

4.2.1 Observations

We participated in the selected projects as an observer to gain a close and intension familiar with the project members and understand the details and processes of the projects. The main idea of this approach is to observe developers performing the activities that they usually do in their daily jobs. To be more specific, observation consists of writing notes about developers' activities, events, interactions, tool usage, and any other phenomena. The digital objects (including source code repository, project documentation, mailing list, code review records, bug reports, and forum) were screened to collect any information related to secure programming. Information collected during the observation was recorded without distracting participants of communities. Observation is an important method to be used in this research because it allowed us to collect information about what learning tools the OSS participants used and how they used them. Moreover, it was a source of valuable insights to assist in a comprehensive understanding of the nature of the case data.

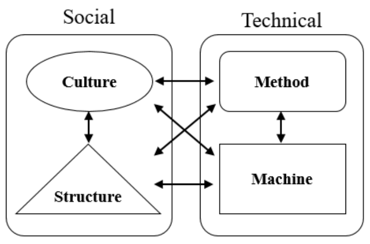
4.2.2. Semi-Structure Interviews

As we wanted to get input from the OSS participants, while still allowing them to think freely to some extent, we chose to use a semi-structured interview as described by May [26]. Individual interviews were conducted with 13 participants in the selected three projects during the observation period. Participants with short (less than one year), medium (between 1 to 3 years) or long (more than 3 years) experience in the open source development were interviewed. Most of the interviewees did not want to disclose their identity and project name, thus, we did not represent their names in the finding. Due to the geographical distribution of the interviewees, all interviews were carried out via online communication software (Skype and Google Hangout). We had to accommodate all the interviewees' constraints in the setting of interviews.

All interviews were recorded and lasted approximately one hour. The questions were used to understand their experiences in OSS development and examine their perception of learning processes about secure programming in their OSS communities. In order to facilitate elaboration, certain possible follow up questions were prepared beforehand. As we suspected that the subjects would be unwilling to consider themselves behaving insecurely, we also asked about what other members would do. This also has the benefit of covering more subjects.

4.3 Data Analysis

Because this study deals with security issues in OSS development phenomenon, including both social and technical aspects, it is important to analyze and model the research context using system-holistic and socio-technical approaches. As Scacchi [27] points out, the meaning of OSS in the socio-technical context is broader than its technical definition and includes communities of practice, social culture, technical practices, processes, and organizational structures. In this study, we adopt a socio-technical system (STS) developed by Kowalski [28] to summarize and classify our findings. The STS model is depicted in Fig. 2. STS provides a structural analysis of the relationship between the social (cultural/ethical, structural/ managerial) and technical elements (method/process and machine /technique) to give a complete overview of the status. Furthermore, the STS model allows the results to be categorized and presented in such a way that they are easy to understand by non-specialists.



(a)

Culture	**Method**
• *Security culture / value*	• *Self-directed learning* • *Learning from mistakes*
Structure	**Machine**
• *Security expertise coordination*	• *Non-unified security knowledge sharing mechanisms*

(b)

Fig. 2. (a) Socio-technical system [18], and (b) socio-technical analysis of research findings

4.3.1. Method category

Method category of STS presents our findings regarding the learning process about secure programming knowledge. Our study firstly found that learning processes of secure programming for OSS developers are centered on informal-based and self-directed learning experiences. The openness and transparency of OSS projects provide an interesting setting for participants to exercise self-directed learning. In OSS development, participants usually first try to solve their problems themselves by the mean of available materials and if required by exploring the web: browsing documentations (guideline, wiki, FAQ, etc.), studying the source code and engaging in discussion threads. These internet resources have the advantage to provide the community with an information infrastructure for publishing and sharing description of software development in the form of hypertext, video, and a software artifact content indexes or directories.

Another learning process about secure programming in OSS communities is learning from mistakes. Our study found that OSS developers care more about making the software work eventually rather than trying to make it work at the very first time. When contributing to the projects, they mostly focus on their immediate goals that usually involve functional requirements and system performances and then patch whatever bugs there may be when it is time for the next release or hotfix. They have not considered the importance of a given function might have to the overall security of their application until they made mistakes and understand the consequence of the flaw. The learning ability from the mistakes become essential in this context. The process of code review is an important enabler for a developer to reflect their code, take corrective actions and build concrete coding experience. Not only members are opened about their mistakes, but they also share their experience as learning opportunities for others.

4.3.2. Structure category

Structure category of STS presents the social elements is managerial and administrative perspectives. Security expertise coordination is the key element that fell into this category. Security expertise coordination is to combine explicit and tacit knowledge in all management and security expert decisions, and to get knowledge moved from individuals within the whole organization between different actors, and from tacit domain to explicit domain and also vice versa [33]. In this study, expertise coordination is manifested in the following two strategies: coordinating organizational structure and security infostructure.

Every member involves in OSS development should be concerned with software security, but it is inefficient to demand each participant taking care of all security aspects. The coordinating organizational structure serves as subject matter experts to ensure that security-related issues receive necessary attention in the community. Some OSS communities utilize security experts to define security requirements and best practices, help perform code reviews, and provides the necessary education for the software development staff. Through this structural mechanism, the security knowledge is able to gain valuable insights from the organization to facilitate strategic decision-making.

In the knowledge sharing process, infostructure serves as a role to provide rules, which govern the exchange between the actors on the network providing a set of cognitive resources (metaphors, common language) whereby people make sense of events on the network [34]. In the context of OSS development, developers meet face-to-face infrequently if at all, and coordinate their activity primarily by means of digital channels on the internet. A proper infostructure can help learners identify the location of the security information, knowing where an answer to a problem, and acquiring as much knowledge as possible

4.3.3. Machine category

A major problem we found is a lack of sufficient as well as efficient knowledge sharing mechanisms for secure programming in OSS communities. Based on the study, the security information is scattered over the community websites (source code, documentation, wiki, forum, conference pages, etc.), and the quantity of transferred knowledge is varied by projects. Finding and learning knowledge about secure programming becomes a key challenge that is highly dependent on the resources the community provides. However, they rarely address the security requirements in documentation to help drive the team to understand the prioritized security needs of the entire project. Newcomers feel that comprehending systems from exploring the website is hopeless, so they as well prefer to start with programming. They lose the learning opportunity about the security requirements of the project and being aware of the possible mistakes they may make in their code.

4.3.4. Culture category

Security culture is the way our minds are programmed that will create different patterns of thinking, feeling, and actions for the security process [35]. As indicated in the Method category of STS, in the context of OSS development, developers enjoy contributing their code to the project for the fulfillment of functional requirements and performances. Secure software development is often left out from developer's heuristics. Subsequently, we observed that security culture and security knowledge sharing interact with each other in the OSS communities. Security culture decides how much security knowledge is disseminated within the community and what knowledge learners can learn; likewise, with more security knowledge sharing, it can help instill the value of security in the community, and the particular learning behaviors and actions can be expected among participants. The security culture backgrounds either at organizational or at the individual level have impacts on the amount of security knowledge transferred within the community, further, affecting participants' learning processes.

4.4 Suggestions from the Study of the Research Domain Problem

Based on the finding of our ethnographic observation in OSS projects, we have come to an outlook of an artifact that can address the explicated problems about security knowledge sharing and learning in OSS communities.

- The artifact should have the possibility to achieve the effectiveness of the learning process about software security in the open source software community by considering different socio-technical perspectives. (Fig. 3).

- The artifact should be as a role of security expertise coordination, organizing and transferring useful security information, establishing rules and norms adaptive to the context of the OSS project.
- The artifact should offer opportunities for learning and self-development of software security knowledge with limited support from security experts or other peers.

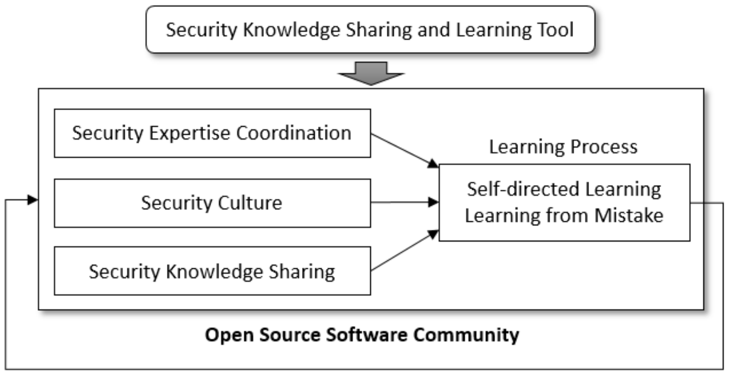

Security Knowledge Sharing and Learning Tool
Security Expertise Coordination
Security Culture
Security Knowledge Sharing
Learning Process
Self-directed Learning
Learning from Mistake
Open Source Software Community

Fig. 3. Improving security learning process by considering different perspectives

5 A Proposed Security Knowledge Sharing and Learning Application

In this section, we present our proposed security knowledge sharing and learning application based on the observations and suggestions from the empirical study. The conceptual design depicted in Fig. 4 defines an abstract view of the application, which deals with the development of an ontology-driven web application for presenting software security knowledge in multiple formats. This application will provide a web-based environment that allows OSS participants or learners adaptively retrieve and present security knowledge content according to the context of the application developed by the project. In the following section, we describe three aspects of the application design: Context-sensitive ontology-based knowledge model, functional architecture, and system architecture of the application.

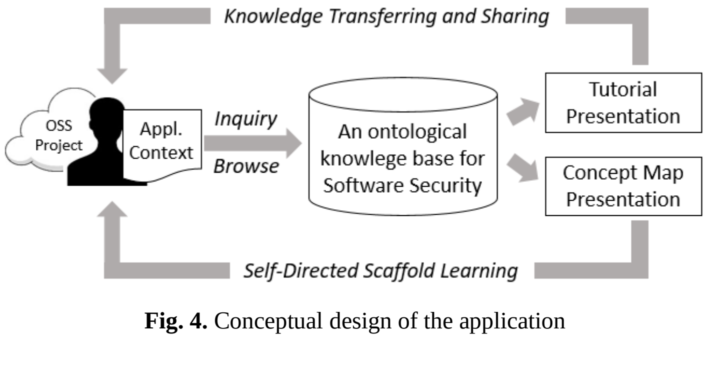

Knowledge Transferring and Sharing
OSS Project
Appl. Context
Inquiry
Browse
An ontological knowlege base for Software Security
Tutorial Presentation
Concept Map Presentation
Self-Directed Scaffold Learning

Fig. 4. Conceptual design of the application

5.1 The context-sensitive ontology-based knowledge model

Ontology facilitates capture and construction of domain knowledge and enables representation of skeletal knowledge to facilitate the integration of knowledge bases irrespective of the heterogeneity of knowledge sources [38]. The basic concept of our ontology design is to provide a vocabulary for representing knowledge about software security domain and for providing linkages with specific situations in the application context. Fig. 5 illustrates a simplified view of the tentative design of the context-sensitive ontology-based model.

In order to effectively regulate the operation of security knowledge and be an essential part of the project knowledge, security knowledge must incorporate additional features. First, there is the requirement of a security domain model, which identifies fundamental entity types and relationship between them. With this model, all concepts of security domain knowledge are described at a level of abstractions, which enables cohesively treating entitles falling under the same conceptualization. Second and most important, knowledge, security knowledge must incorporate context, that is, to be modeled with certain characteristics of applications, such as software paradigms, programming languages and used technologies. The contextual information acts as a filter that defines, in a given context, what security knowledge pieces must be taken into account.

The main advantage of this ontology model is to share a common understanding of the structure of security knowledge along different SDLC activities and among different software development context to enable semantic interoperability. With this design, software engineers are allowed to find solutions to exceptional situations by searching for similar context. For example, a PHP web application designer can refer to other projects' security setup by looking for the same domain (subject area) and software technologies. The major terms used in the security domain ontology are explained in Table 1 while Table 2 lists the characteristics in the application context model.

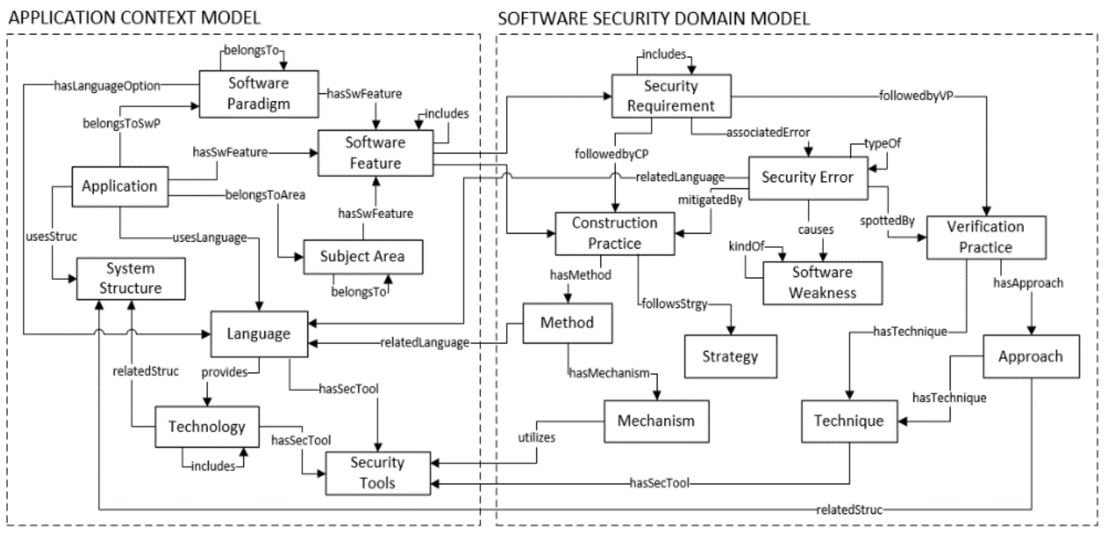


Fig. 5. A tentative design of the ontology-based context model

Table 2. The definition of major terms in security domain ontology

Class	Definition
Security Requirement	A software security requirement can be defined as a software requirement needed to avoid a specific software security error during the development [43].
Construction Practice	Construction practices focus on proactive activities for an organization to design and build secure software by default [44].
Verification Practice	Software verification is to assure that software fully satisfies all the expected requirements [44], which typically include code review practices and security testing practices. Knowledge elements in verification practices are organized in two catalogs: Technique and Approach.
Security Error	A software security error is a tangible manifestation of a mistake in the software development artifacts of a piece of software that causes a software weakness [43, 45].
Software Weakness	Software weaknesses are flaws, faults, bugs, vulnerabilities, and other errors in software implementation, code, design, or architecture that if left unaddressed could result in systems and networks being vulnerable to attack [46].

Table 3. Identified characteristics (classes) in the application context

Class	**Definition**	**Example**
Software paradigm	It represents the categories of software applications that share common development characteristics.	Web application, desktop application, mobile application
Subject area	It represents domains that an application belongs.	Banking, health, Travel
Software feature	It represents the essential elements of software with security concerns. The software feature is associated with the software paradigm and the subject area of the application.	User authentication, credit card processing, file upload.
Language	It represents programming language used to develop an application.	C, C++, Java, JavaScript, PHP
Technology	It represents the combination of frameworks or tools used to create an application. Technologies are built based on programming languages.	Web framework (Symphony for PHP, Angular for JavaScript, etc.) toolkit, SDK
System structure	It represents the fundamental structure to operate the application.	Database management system, runtime platform (MS Windows, Android)
Security Tool	It represents a concrete solution to implement construction mechanisms or verification techniques.	HTML Purifier, PHPUnit for PHP testing

5.2 Functional Architecture

The functional architecture (Fig. 6) illustrates the functionalities supported in the application, which are divided into two categories: knowledge presentation modules and system management modules. The knowledge representation module is to provide learners with learning materials. By making better use of the domain knowledge and contextual information, it is planned that the optimal materials are provided. The system management module is responsible for maintaining the ontology repository and loading knowledge content for the needs of the knowledge presentation. We give a brief for each module in below sections

5.2.1 Query Module

Users can access the security knowledge through a query interface passing requests to a search engine. The input criteria can be constructed dynamically or use pre-configured question patterns.

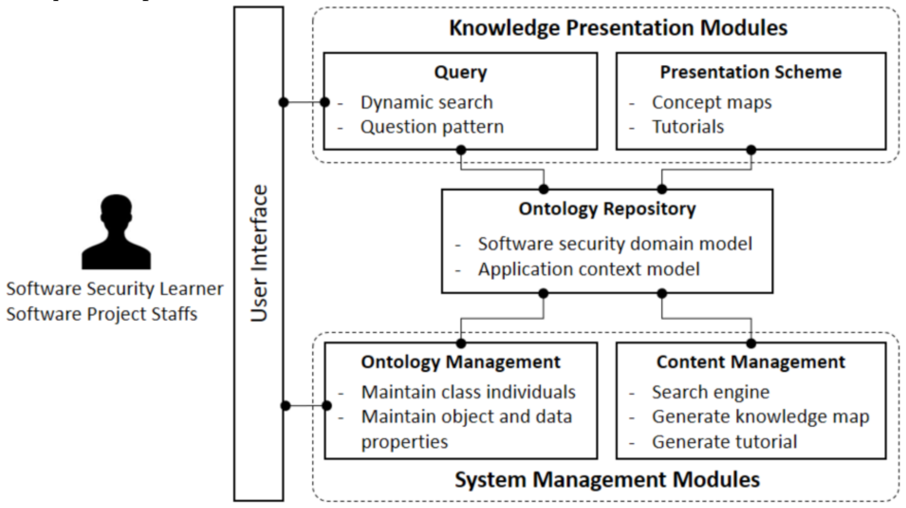


Fig. 6. The functional architecture of the proposed application.

5.2.2 Presentation scheme module

Regarding the knowledge presentation, two knowledge representation schemes are suggested in order to provide a navigation view and integrated information view of the knowledge items: concept maps and tutorials. Concept maps are the graphical representation of the knowledge items and their relationship. Concept maps can be used as primary sources for knowledge acquisition, adjunct aids to text processing, communication tools for organizing ideas, or retrieval cues [47]. In tutorials, knowledge content is presented as a form of static web pages, which are allowed users to browse the detailed information and relevant resources, such as sample code or URL links.

Ontology management module.

Ontology management module is responsible for maintaining and loading data from the ontology, including specific individuals of classes, object properties, and data properties.

Content management function.

Content management module acts as services to receive requests from users, to interact with ontology management functions, and to process and display the result set according to the requested knowledge presentation format.

5.3 Technical Architecture

Fig. 7 provides an overview of our proposed architecture implementing the main feature of the application outlined above. The front-end has been designed as JSP pages and through them, the users can access the various modules and functions of the application. Clients can interact with the server (Apache Tomcat) using an HTTP request to a Java Servlet. The backend is implemented in Java and access to the ontology repository is provided through the Jena API[1], 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[2]. Pellet[3] is an open source OWL DL (descriptive logic) for Java, which is used to infer relevant knowledge from the ontology defined in the OWL. Pellet can also be integrated with Jena or OWL API libraries.

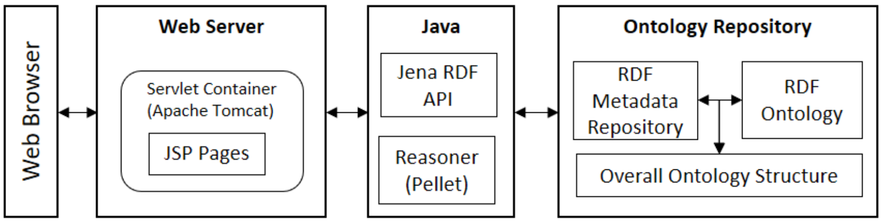


Fig. 7. The system architecture of the ontology-based web application.

6 Research Steps

Our research adopts Design Science Research (DSR) methodology, presented in section 3, to form a development framework for our proposed artifact, a web-based security knowledge sharing and learning system in the OSS community. As the future work, we intend to carry out activities to facilitate the artifact development works using three DSR iterations (See Fig. 8). The detailed research steps are described in the following.

[1] https://jena.apache.org/
[2] https://www.w3.org/2001/sw/
[3] https://www.w3.org/2001/sw/wiki/Pellet

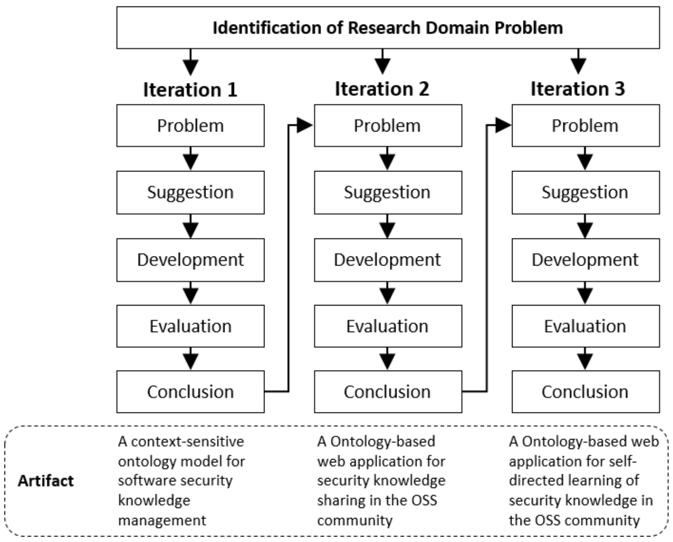

Identification of Research Domain Problem
Iteration 1
Iteration 2
Iteration 3
Problem
Suggestion
Development
Evaluation
Conclusion
Artifact
A context-sensitive ontology model for software security knowledge management
A Ontology-based web application for security knowledge sharing in the OSS community
A Ontology-based web application for self-directed learning of security knowledge in the OSS community

Fig. 8. Design science research iterations.

Iteration 1: Create a context-sensitive ontology for software security knowledge management.

- Awareness of the problem (Proposal): There is a gap between knowledge available and knowledge required to build secure software system in the context of OSS development.
- Suggestion (Tentative design): Software security knowledge should be modeled and managed in a context-sensitive manner where the software security knowledge can be retrieved taking the context of the OSS application in hand into consideration.
- Development (Artifact): Design of the new ontological knowledge model for software security knowledge management using Protégé OWL tool [48].
- Evaluation: To demonstrate the results and possibility from the development of the ontology, three web application vulnerabilities data will be modeled (Cross-site scripting, SQL injection and Cross-site request forgery). A compatible tool, SPARQL, to infer and to query on the ontology in OWL standard will be adopted.

Iteration 2: To create an ontology-based web application for security knowledge sharing in OSS communities.

- Awareness of the problem (Proposal): There lacks a unified security knowledge sharing tool in OSS communities.
- Suggestion (Tentative design): Design a web application with the proposed context-sensitive ontology, which supports distributing the necessary security knowledge within the OSS community.
- Development (Artifact): To implement the web application with the proposed application architecture presented in Figure 6.

- Evaluation: Two experiments will be taken in this iteration:
    - The artifact will be tested with a number of individuals/students in academic settings in order to get feedback about system usability. The questionnaire will be designed to get the conception of the experiment subjects about the artifact.
    - The artifact will be tested in practices with members in 2 selected OSS communities. This experiment aims to testify the feasibility of security knowledge sharing using the artifact.

Iteration 3: To enhance the artifact from iteration 2 and provide features of self-directed learning for members in OSS communities.

- Awareness of the problem (Proposal): There exist different levels of security skills and experience among participants in the OSS community. The transferred security knowledge should be adaptive to the level of security knowledge of learners.
- Suggestion (Tentative design): The learning content should also be applicable to management and security readership more generally and should be appealing to all those who are concerned about software security in OSS communities.
- Development (Artifact): To adopt Concept Map in scaffold learning of software security knowledge.
- Evaluation: The artifact will be tested in practices with members in 3 selected OSS communities. The level of security knowledge is the independent variable. Questionnaires will be designed to test the level of security knowledge before and after using the artifact.

7 Conclusion and Future Works

In this paper, we present our ongoing research work in the field of security knowledge management in OSS communities. Learning software security is a difficult and challenging task since the domain is quite context-specific and the real project situation is necessary to apply the security concepts within the specific system. Identifying security knowledge that is applicable in a given context can become a major challenge for OSS participants. OSS communities cannot expect all its participants to be educated in software security before joining the project, and therefore must take some responsibility for educating both developers and users on potential security errors and the relevant mitigations. Although some OSS communities do have security experts and built-in security practices, many others do not. As a result, the issue of how to support developers or other learners reaching the required level of security knowledge to secure OSS development becomes an important topic.

As the future work, we intend to implement the proposed intelligent application and to evaluate its usability in educational paradigms and software development projects. Our ultimate goal is to provide open source software communities, a set of advanced services for efficiently handling and disseminating software security knowledge within the community.

References

[1] Humes, L.L. (2007), "Communities of Practice for Open Source Software", in Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives, IGI Global. pages 610-623.

[2] Scacchi, W., et al. (2006), "Understanding free/open source software development processes". Software Process: Improvement and Practice, volume 11, issue 2, pages 95-105.

[3] Feller, J. and B. Fitzgerald (2002), "Understanding open source software development". volume: Addison-Wesley London.

[4] Feller, J., et al. (2006), "Developing open source software: a community-based analysis of research", in Social Inclusion: Societal and Organizational Implications for Information Systems, Springer. pages 261-278.

[5] NorthBridge, B., "2016 Future of Open Source Survey", Electronic document. http://www.northbridge.com/2016-future-open-source-survey-results.

[6] BlackDuck Software, "2017 Open Source Security and Risk Analysis", Web: https://www.blackducksoftware.com/open-source-security-risk-analysis-2017.

[7] Wen, S.-F. (2017), "Software Security in Open Source Development: A Systematic Literature Review". in Proceedings of the 21st Conference of Open Innovations Association FRUCT. Helsinki, Finland.

[8] Pittenger, M. (2016), "Know your open source code". Network Security, volume 2016, issue 5, pages 11-15.

[9] Levy, J., "Top Open Source Security Vulnerabilities", WhiteSource Blog. Accessed 22 Jun 2018. https://www.whitesourcesoftware.com/whitesource-blog/open-source-security-vulnerability/.

[10] Agrawal, A., et al. (2017), "We Don't Need Another Hero? The Impact of" Heroes" on Software Development". arXiv preprint arXiv:1710.09055.

[11] Benbya, H. and N. Belbaly (2010), "Understanding developers' motives in open source projects: a multi-theoretical framework".

[12] Jaatun, M.G., et al. (2011), "A Lightweight Approach to Secure Software Engineering". A Multidisciplinary Introduction to Information Security, volume, issue, pages 183.

[13] McGraw, G. (2006), "Software security: building security in". volume 1. Addison-Wesley Professional.

[14] Apvrille, A. and M. Pourzandi (2005), "Secure software development by example". IEEE Security & Privacy, volume 3, issue 4, pages 10-17.

[15] Wen, S.-F. (2016), "Hyper Contextual Software Security Management for Open Source Software". in STPIS@ CAiSE.

[16] Mead, N.R., et al. (2004), "Software security engineering: a guide for project managers". volume: Addison-Wesley Professional.

[17] Viega, J. and G.R. McGraw (2001), "Building secure software: how to avoid security problems the right way".

[18] Xie, J., H.R. Lipford, and B. Chu (2011), "Why do programmers make security errors?". in Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on. IEEE.

[19] Graff, M. and K.R. Van Wyk (2003), "Secure coding: principles and practices". volume: " O'Reilly Media, Inc.".

[20] Birkenkrahe, M. (2002), "How large multi-nationals manage their knowledge". Business Review, volume 4, issue 2, pages 2-12.

[21] Vaishnavi, V. and W. Kuechler (2004), "Design research in information systems".

[22] Von Alan, R.H., et al. (2004), "Design science in information systems research". MIS quarterly, volume 28, issue 1, pages 75-105.

[23] Sharp, H., Y. Dittrich, and C.R. de Souza (2016), "The role of ethnographic studies in empirical software engineering". IEEE Transactions on Software Engineering, volume 42, issue 8, pages 786-804.

[24] Baxter, G. and I. Sommerville (2011), "Socio-technical systems: From design methods to systems engineering". Interacting with computers, volume 23, issue 1, pages 4-17.

[25] Kuhn, D.R., M. Raunak, and R. Kacker (2017), "An Analysis of Vulnerability Trends, 2008-2016". in Software Quality, Reliability and Security Companion (QRS-C), 2017 IEEE International Conference on. IEEE.

[26] May, T. (2011), "Social research". volume: McGraw-Hill Education (UK).

[27] Scacchi, W. (2002), "Understanding the requirements for developing open source software systems". in IEE Proceedings--Software. IET.

[28] Kowalski, S. (1994), "IT insecurity: a multi-discipline inquiry", PhD Thesis, Department of Computer and System Sciences, University of Stockholm and Royal Institute of Technology, Sweden. ISBN: 91-7153-207-2.

[29] Al Sabbagh, B. and S. Kowalski (2013), "A socio-technical framework for threat modeling a software supply chain". in The 2013 Dewald Roode Workshop on Information Systems Security Research, October 4-5, 2013, Niagara Falls, New York, USA. International Federation for Information Processing.

[30] Bider, I. and S. Kowalski (2014), "A framework for synchronizing human behavior, processes and support systems using a socio-technical approach", in Enterprise, Business-Process and Information Systems Modeling, Springer. pages 109-123.

[31] Karokola, G., L. Yngström, and S. Kowalski (2012), "Secure e-government services: A comparative analysis of e-government maturity models for the developing regions–The need for security services". International Journal of Electronic Government Research (IJEGR), volume 8, issue 1, pages 1-25.

[32] Wahlgren, G. and S. Kowalski (2014), "Evaluation of Escalation Maturity Model for IT security risk management: A design science work in progress". in The 2014 Dewald Roode Workshop on Information Systems Security Research, IFIP WG8. 11/WG11. 13. IFIP.

[33] Anttila, J., et al. (2007), "Fulfilling the needs for information security awareness and learning in information society". in The 6th annual security conference, Las Vegas.

[34] Pan, S.L. and H. Scarbrough (1999), "Knowledge management in practice: An exploratory case study". Technology Analysis & Strategic Management, volume 11, issue 3, pages 359-374.

[35] Al Sabbagh, B. and S. Kowalski (2012), "Developing social metrics for security modeling the security culture of it workers individuals (case study)". in Communications, Computers and Applications (MIC-CCA), 2012 Mosharaka International Conference on. IEEE.

[36] Gruber, T.R. (1993), "A translation approach to portable ontology specifications". Knowledge acquisition, volume 5, issue 2, pages 199-220.

[37] Wand, Y., V.C. Storey, and R. Weber (1999), "An ontological analysis of the relationship construct in conceptual modeling". ACM Transactions on Database Systems (TODS), volume 24, issue 4, pages 494-528.

[38] Gruber, T.R. (1995), "Toward principles for the design of ontologies used for knowledge sharing?". International journal of human-computer studies, volume 43, issue 5-6, pages 907-928.

[39] Uschold, M. and M. Gruninger (1996), "Ontologies: Principles, methods and applications". The knowledge engineering review, volume 11, issue 2, pages 93-136.

[40] Noy, N.F. and D.L. McGuinness (2001), "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, CA.

[41] Wang, X., et al. (2004), "Semantic space: An infrastructure for smart spaces". IEEE Pervasive computing, volume 3, issue 3, pages 32-39.

[42] Gruninger, M. (2002), "Ontology: applications and design". Commun. ACM, volume 45, issue 2.

[43] Khan, M.U.A. and M. Zulkernine (2008), "Quantifying security in secure software development phases". in Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International. IEEE.

[44] Chandra, P. (2009), "The Software Assurance Maturity Model-A guide to building security into software development".

[45] Landwehr, C.E., et al. (1994), "A taxonomy of computer program security flaws". ACM Computing Surveys (CSUR), volume 26, issue 3, pages 211-254.

[46] MITRE, "Common Weakness Enumeration, Frequently Asked Questions"; Available from: https://cwe.mitre.org/about/faq.html#A.1.

[47] O'donnell, A.M., D.F. Dansereau, and R.H. Hall (2002), "Knowledge maps as scaffolds for cognitive processing". Educational psychology review, volume 14, issue 1, pages 71-86.

[48] Tudorache, T., et al. (2013), "WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web". Semantic web, volume 4, issue 1, pages 89-99.