ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32

https://doi.org/10.1145/3178158.3178202

Learning Secure Programming in Open Source Software Communities: A Socio-Technical View

Shao-Fang Wen
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology,
shao-fang.wen@ntnu.no

ABSTRACT

In open source software (OSS) communities, volunteers collaborate and integrate expertise to develop the software online via the Internet in a decentralized, highly interactive and knowledge-intensive process. Development of qualified and secured software products relies mainly on the ability of OSS participants to acquire, refine and use new aspects of secure programming knowledge. Many OSS proponents believe that the open source innovation offers significant learning opportunities from its best practices. However, studies that specifically explore learning of software security in the context of open source development are scarce. This paper aims to empirically assess present knowledge sharing and learning about secure programming knowledge in the context of OSS communities utilized a socio-technical approach on OSS projects based on an ethnographic observation. Our motivation 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 learning processes of secure programming in OSS communities.

CCS Concepts

•Social and professional topics → Project and people management.

Keywords

Open source software; open source software community; secure programming; software security; socio-technical

1. INTRODUCTION

Open source software (OSS) is based on the principle that computer programs should be shared freely among users, giving them the possibility of introducing improvements and modifications. OSS is at the core of today's IT infrastructure and information systems: about 80% of companies run their operations on OSS [33] and 96% of applications utilize OSS as the software components [6]. OSS security has been the focus of the security community and practitioners over the past decades. Many studies have been conducted by both researchers and practitioners on the mechanisms of building security in OSS development [45]. However, the number of new vulnerabilities keeps increasing in today's OSS systems. The Blackduck 2017 Open Source Security and Risk Analysis report has announced that 3623 new OSS vulnerabilities occurred in 2016 – almost 10 per day on average and a 10% increase from 2015 [6]. According to 2017 NIST report, about 67% of vulnerabilities are due to programming errors; the rest are due to configuration or design problems [24]. In particular, a strong majority has been found to be classic errors that are fairly well known, like buffer overflows, cross-site scripting and injection flaws. With today's increasing importance and complexity of OSS, the ineffective learning of knowledge and skills relevant to secure programming practices in OSS development will result in more breaches that are serious in the future.

Learning secure programming is a difficult and challenging task since the domain is quite contexted specific, and the real project situation is necessary to apply the security concepts within the specific system. In the context of OSS, development of qualified and secured software products relies mainly on the ability of developers to acquire, refine and use new aspects of secure programming knowledge in their communities. Many OSS proponents believe that the OSS community offers significant learning opportunities from its best-practices [17, 25], which are different from the education of traditional model [7, 13]. However, studies that specifically explore learning of secure programming in OSS communities are scarce.

On the other hand, OSS is developed collectively by the online community of practices with a strong relationship between the technical and social interactions in a knowledge-intensive process [16, 21]. As Scacchi [36] points out, the meaning of open source in the socio-technical context is broader than its technical definition and includes communities of practice, social culture, technical practices, processes, and organizational structures. This can be viewed as a necessary condition within a learning framework as both aspects are of equal importance [29]. Technical learning mechanisms considering different social aspects (e.g., organizational culture and structure) of OSS development will assure the effectiveness and efficiency of the learning process.

Given the background, this study was designed to empirically assess present knowledge acquisition and learning about secure programming in OSS communities utilized a socio-technical approach on OSS projects based on an ethnographic observation. 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 secure programming 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 learning processes of secure programming in OSS communities. The rest of this paper is structured as follows. Section 2 describes the

ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32

https://doi.org/10.1145/3178158.3178202

literature review, including learning in open source communities and the views on socio-technical aspects. The research method is explained in Section 3. In section 4, we present the result of data analysis. Section 5 provides a discussion based on the result. Section 6 states the limitation of this study. Finally, we describe the conclusion in section 7.

2. LITERATURE REVIEW

2.1 Learning in Open Source Communities

In open source software (OSS) communities, volunteers collaborate and integrate expertise to develop applications and solve particular programming problems via the Internet in a decentralized, highly interactive and knowledge-intensive process [16, 21]. Larger numbers of technical and non-technical users get participation in activities that are essential for the OSS development process, as well as the maintenance and diffusion of the software [11, 12, 37]. The activities that these communities perform are usually called OSS projects, in which the software source code is freely available on repositories on the internet. A OSS community has been considered as a virtual (online) community of practice (CoP) [15, 32, 42] which aims to establish a structure where tacit and explicit knowledge is shared and exchanged among various members within a given domain to create a collective value useful to everyone [27, 46]. Developers work on projects that interest them and by so doing, they acquire knowledge associated with their profession. OSS communities offer 24 hours, 7 days a week, 365 days support with up to date content and learning materials, and all of this provided by volunteers at no charge. Therefore, an open source community is more than about software development, but also provides a rich field to explore the process of software knowledge creation, accumulation, and dissemination [42].

Learning in open source communities have been broadly studied in the literature. Hemetsberger and Reinhardt [15, 16] examined how knowledge sharing and learning processes develop at the interface of technology and communal structures of an OSS community. They suggested that knowledge is shared and learned in OSS communities through the establishment of processes and technologies that enable virtual re-experience for the learners at various levels. They viewed learning in OSS communities as experiential learning whereas learning is a process whereby learning is created through the transformation of experiences as developed by Kolb [22]. Au et al. [4] explored open-source debugging as a form of organizational learning, which heavily relies on adaptive learning [44] to overcome the complexity of software. Singh and Holt [41] provided insights on how the OSS community uses the forums for learning and solving problems. They explored the motivations for joining OSS communities, the learning that occurs in the communities, and the challenges to learning. Hardi [14] had a case study using Google Chrome project to affirm that situated learning [26] is present among open source developers at an earlier time of a project. Although rapidly growing the current number of studies on learning in OSS communities, the study on the fields of software security is still limited.

2.2 The Views on Socio-Technical Aspects

Software systems are not purely technical objects. They are designed, constructed and used by people. Therefore, software systems are components in socio-technical systems, which include technological as well as social structures. The socio-technical aspects provide a deeper analysis of the relationship between the methods, techniques, tools, development environments and organizational structures [20, 21]. There are more and more literature containing applications of the socio-technical systems of software engineering. For example, Lu and Jing [28] present a socio-technical approach to support integrated socio-technical negotiation activities in a collaborative software design process. They address the critical issues of such collaborative negotiation activities, including modeling negotiation arguments based on social and technical factors and analyze these arguments to reconcile the conflicts for software design tasks. Ducheneaut [10] examines socialization of new members in an open source community using socio-technical analysis since these members interact with both people and material components of a project. Ye et al. [47] propose a socio-technical platform to guide the design of software that supports information seeking and communication during different phases of programming.

Our research is based on the theory of Socio-Technical System (STS) developed by Kowalski [23]. The STS model is depicted in Figure 1. This model has two sub-systems include social aspects (culture and structures) and technical aspects (methods and machines). STS model has been applied to evaluate threat modeling in software supply chain [2], business process re-engineering [4], a framework for securing e-Government services [19] an information security maturity model [21]. The STS provides an appropriate and legitimate way to perform system analysis through a systemic–holistic perspective and help us understand the intrinsic context in open source phenomenon.

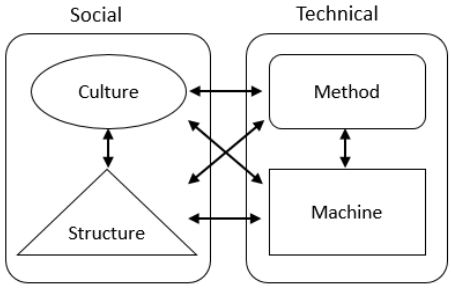

Figure 1. Socio-technical system [23]

ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32
https://doi.org/10.1145/3178158.3178207

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++

3. METHODOLOGY

The research methodology used for this empirical study on OSS communities can be characterized as a qualitative research inspired by ethnography. An ethnographic approach typically includes fieldwork done in natural settings, the study of the larger picture to provide a complete context of the activity, an objective perspective with rich descriptions of people, environments, and interactions, and an aim toward understanding activities from the informants' perspective. In empirical software engineering, ethnography provides an in-depth understanding of the socio-technical realities surrounding everyday software development practice [40] and highlights the significance of socio-technical issues in the design of software-intensive systems [5]. The knowledge gained can be used to improve processes, methods, and tools as well as to advance the observed security practices.

3.1 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.

3.2 Data Collection

In terms of data collection, two methods used with qualitative data collection were adopted: observation and interviews.

3.2.1 Observation

The author of this paper 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.

3.2.2 Semi-Structure Interviews

As we wanted to get input from the OSS participants, while still allowing for them to think freely to some extent, we chose to use a semi-structured interview as described by May [30]. 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. DATA ANALYSIS

4.1 Brief of Case Observations

4.1.1 Project A

Project A is an online text editor that allows real-time, collaborative text editing started from mid of 2014. The project website provides basic documentation about what the software does, how to install the software and how to contribute to the project. Security-related information (e.g., coding style, security contact window, etc.) has not yet been published. During the study, it was found that one of the core developers was responsible for security tasks, who solely outlined the security strategy, and had implemented a major portion of security requirements, such as privilege management, user certification, and output control. The source code is routinely refactored in a well-structured format and detailed comments are given in program segments, which is also embedded critical security designs.

Google Group is used as the discussion forum in the community. No discussion threads about software security were observed until the time we studied. There were four bug reports related to the cross-site scripting vulnerabilities, which were labeled as 'security' or 'xss' along with developers' discussions and the final code commits. The project held a weekly virtual meeting (Google hangout) to discuss project strategies and roadmaps, and the security issues were brought up during the discussion intermittently Most of the time the attendees were the core developers. The virtual meetings were not recorded, and no documentation was made after meetings. Facebook and Twitter are used to announce new features and release updates.

ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32

https://doi.org/10.1145/3178158.3178202

4.1.2 Project B

Project B has been established for 8 years, which aims to provide a web content management system for building robust, flexible websites. Their approach to security is primarily focused on writing quality code, with the objective being to making security a priority. For the purpose, in 2012, they formalized a security team with six official members to coordinate the security tasks of the project. The security team published various documentation to educate the community on security best practices and their development process, including secure coding practices, security risk assessment and peer review to help ensure the products are high quality and secure. In their secure coding practices, for example, they introduce how to sanitize text to avoid improper neutralization input during web page generation (cross-site scripting) and how to use the provided database function to access the database to guard against SQL injection attacks, etc. The security issue is taken seriously in the project. To protect the security of their services, all security issues about the product must be reported directly to a specific email address using PGP encryption. Emails sent to this address are forwarded to the security team private mailing list. After evaluating potential impact and correcting the vulnerability, the security team discloses the security issue with a security advisory.

The project has a set of communication channels to cover different participants and community needs, including mailing lists, IRC chatting, discussion forum, Blog, social media (Facebook and Twitter), and yearly face-to-face meetings. To efficient spread security information, a dedicated mailing list is set up for secure communication among users and developers, and a separate channel on IRC for security. The blog is an important medium to share security information with the community, which contains numerous technical documents and papers about the project that are given by the developers, users, and sponsors. The face-to-face meetings are 2 days events that follow a conference format. The lectures given at conferences were all recorded and provided in conference pages.

4.1.3 Project C

Project C is a multimedia playback engine across browsers and media types, which is a community-driven free software effort focused on delivering a high-performance and reliable music player. The project web pages contain answers to frequently asked questions (FAQ) about the software, such as how to build and install software from source and what steam types that the software supports, etc. A wiki website is a setup mainly for introducing the development works, including IDE (Integrated Developer Environment) setup, coding guidelines, language bindings, and APIs (Application Programming Interfaces). Since their application may handle data from variable sources that might be possibly untrusted, the software security is considered highly critical by the project team. To support security tasks, the project has recruited external security auditors who are responsible for reviewing source code to discover potential security weaknesses, bugs, and violations of programming conventions. Once a bug with potential exploitation is found during the auditing process, the auditing team will coordinate the code owner to handle the bug. The bug fix information will then be posted to the security announcement page, and copies will be sent to the project announce mailing list. The auditing team also acts as a committed reviewer who is committed to the overall quality and correctness of the pull request (submitted code) in GitHub. The most common vulnerability found in the project software is 'Buffer overflow (stack-based and heap-based)', which allows remote attackers to execute arbitrary code via a crafted media file.

Regarding other communication mechanisms within the community, the links of all possible channels are clearly organized in one web page. The mailing list is the main discussion channel within the community. Several mailing lists have been set up for different types of audiences and purposes (e.g. development topics, users topics, announcement, etc.). The project also establishes a question and answer (Q&A) web platform using Stack Overflow, which is mainly for new contributors and software users. For live chatting, they are using Slack for the teams' instant communication tool. There have no questions about software security been asked or discussed in Q&A or in Slack. The virtual meeting is hosted every two weeks using Bluejeans where the status update of code review is on the routine agenda. Many video-based learning materials were made by the community member and placed in YouTube project channel.

4.2 Arenas for Learning

Opportunities for learning secure programming are not only dependent on the initiative of the learner and the response of other community participants, but also on the arenas where learners meet, communicate and act together. Since OSS communities are all hosted on the internet, their project websites play the central arena that affords learning opportunities.

4.2.1 Exploring Project Knowledgebase

In our study, all the three communities use a mixed method to host their project data: the project website and GitHub. GitHub facilitates social coding by providing a web interface to the code repository ("Git") and management tools for collaboration. Project website provides more flexibility in communication within the community, such as Blog, forum, conference page, etc.

Documentation provides OSS participants with a shortcut for obtaining an overview of the system or for understanding the code that provides a particular feature. At the very least, it includes instructions on how to get started and details of where to find more information. In our observation, documentation about security knowledge scattered over the community websites. In project B, documentation covers a wide range of security perspectives of the project needs (secure coding practice, risk assessment, etc.) while project A and C provide only information for software installation and development guidance. One participant from Project B noted that the documentation "had me backing up and restarting several security concepts in web applications, [since] my programming experience is limited to the desktop environment."

Both Project B and C provided video-based learning materials for participants (recorded from a conference or homemade video) in project website or YouTube respectively. Watching video is viewed stand-alone from other forms of training requiring no interaction from the learners. With the explosion of video-sharing services such as YouTube, the amount of recorded audiovisual information has grown exponentially in open source communities. Respondents gave opinions about learning software security from watching the videos:

"It [Video] allows me to attend the lecture on a flexible schedule and move at my own pace."

ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32

https://doi.org/10.1145/3178158.3178202

It is noteworthy that some technical talks with security topics in the conference are recorded and provided on YouTube, however, few respondents claimed that they reached the video and learned about secure programming from it.

"…It costs me much time to watch the conference video full of contents"

"It is good there is transcription. I can search [keyword] on it."

Reading release notices (security advisories) can be an opportunity for exploring security knowledge. A security advisory is a way for open source communities to communicate security information to the public. Usually, it involves updating to a new release of the code that fixes the security problem. From reading the security advisory, learners can learn not only security enhancements or changes that are related to security vulnerabilities but guidance and mitigations that may be applicable for publicly disclosed vulnerabilities. In Project B, the security-related issues are kept secret until the advisory is ready to be released, at which point it is publicized widely so that all developers and users can address it quickly. A respondent indicated:

"Because it was short, to the point, and very accountable"

4.2.2 Reading Source Code

The source code is seen as the actual documentation while the other kinds of documentation are informally produced to support situated discussion. Reading source code is a key activity in OSS maintenance. Developers can profitably apply experiences and reading systems from text databases. Most interview respondents agree that studying source code of the project expands the horizons of software security. The respondents stated that:

"For me, reading source code is about learning new strategies for solving security problems."

"If you check the PHP code of the Symfony framework, for example, you will know about dependency injection, events, the model/view/controller pattern, and so on."

It is found that only code authors have intentions to highlight what they have done about security in their code via medialization or enough commenting, code reader hardly learn software security from it. Wide gulf in knowledge and experience between security experts and beginners can be a barrier to learn software security efficiently from source code. Some respondents expressed the difficulties in learning security from source code. Some comments are collected:

"It would take 1 to 2 years [for new contributors] to capture the coding patterns and algorisms…to understand meanings behind the code."

"When I was a junior developer I used to lament the lack of comments in code created by team members. Now that I am a lot more experienced, many comments tend to clutter up the code and reduce the comprehensibility."

"If you're a first-time web developer and check on GitHub repository saying something like "CSRF vulnerability", it can often be hard to tell what exactly that is, what impact it has on your application, and how to fix it.."

From project management perspectives, since the new code is merged into the main codebase irregularly, to keep the source code in a good level of readability is a challenge to OSS maintenance. One respondent of Project A commented:

"We have to spend extra time doing codebase refactoring and commenting, to make the code cleaner, simpler and readable."

4.2.3 Following mailing list and forum

Mailing lists and forums are common places for community communications to discuss requirements of the software or development issues, and they are also the main places to provide support to users. These asynchronous communication technologies are not only valuable for knowledge creation purposes, but also in order to make community members think before they act and respond. The respondents commented about mailing list and forum (Project B & C):

"Many of my problems are solved by just browsing through other people's question."

"…more than questions and answers. The messages contain the path leading to answers to the question."

"Though if you look at mailing lists the project has an extensive discussion about the security architecture whenever somebody has a problem or wishes to change something."

Some respondents expressed they felt a bit frustrated with the gap between the community expertise and their own. They either hesitate to ask questions or cannot catch up the pace of discussions.

"Once you have enough knowledge, it can help."

Furthermore, threaded discussions could be a barrier to secure programming learning, especially for the mailing list, as indicated by respondents:

"If you subscribe a very active list, you could easily pile up several hundred emails in a day."

"A lot of irrelevant information to sift through."

In Projects B, although a mailing list is set up specifically for 'security', it is mainly for the vulnerability reporting. Such mailing list cannot be subscribed by participants, and can only be accessed by the dedicated members.

ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32

https://doi.org/10.1145/3178158.3178202

4.2.4 Engaging in code review

Although in OSS development, a programmer may write a complete program independently from other programmers, the software component that will be still examined by other software engineers. Pull requests (PRs) and coding review represents a form of learning processes in which knowledge is created collectively in a distributed work process. It might shortly address the role of PRs as being a workflow mechanism for a developer to notify team members that a feature or fix, developed on a separate branch, is ready. This lets everybody involved know that he (or she) can review the code, providing a forum discussing the implementation of the proposed feature. Questions, answers, and discussions about the issues are communicated back and forth between the community and the members. The respondents had following comments about code review (pull requests):

"I learned [about security] in this project after constantly getting feedbacks on my pull requests regarding how this kind of code can go wrong."

"It [code review] is an effective learning process for me – a chance to see what mistakes I made and the bad habits that I had usually."

5. DISCUSSION

We summarize and classify the research findings based on the STS model presented in section 2.2. Figure 2 gives an overview representation of the socio-technical model in this study. In the following sections, we give a detailed discussion of each part.

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

Figure 2. Socio-technical analysis of findings

5.1 Self-Directed Learning

Our study found that learning processes of secure programming for OSS developers are centered on reactive and self-directed learning experiences. OSS learners may want to learn according to what provokes their curiosity instead, and that may mean starting from the middle and proceeding to pick out material in order of interests or the level of competency instead of what is being defined by a typical structure. The learning journeys they experience are usually unstructured and non-linear. 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.

5.2 Learning from Mistakes

OSS developers care more about making the software work eventually rather than trying to make it work the very first time. When contributing to the projects, they mostly focus on their immediate goals that usually involve functional requirements and system performances. As Daniela et al. [34] point out that software vulnerabilities are blind spots in developers' heuristics in their daily coding activities. 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 processes of pull requests and code review, for example, are important enablers for developers to reflect their code, take corrective actions and build concrete experiences, most importantly, learn from the mistakes. As noted by one respondent:

"We cannot write code properly, so we need someone to pair with us to smooth our failures."

Subsequently, not only members are opened about their mistakes, they share their experience as learning opportunities for others. This is also helpful for those who have not yet suffered through the same mistakes on the road. Researchers have also indicated that engagement with mistakes fosters the secondary benefits of deep discussion of thought processes and exploratory active learning [31, 39]. When the correct answer is made to the mistakes, though, and people appreciate that the answer is correct as well as why that answer is correct, they are able to integrate that information into memory and improve performance [3].

5.3 Non-Unified Security Knowledge Sharing Mechanisms

A major problem we found is a lack of sufficient as well as efficient knowledge sharing mechanisms for secure programming in OSS communities. As our study, the security knowledge is scattered over the community websites (source code, documentation, wiki, forum,

ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32

https://doi.org/10.1145/3178158.3178202

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. For example, to keep all the code in the repository is in a consistent style, OSS projects normally publish their own coding guideline, a set of conventions (sometimes arbitrary) about how to write code for that project. However, they rarely to address the security requirements in documentation to help drive the team to understand the prioritized security needs of the entire project. Some security talks embedded in the recorded video (conference or learning materials) without proper indication (resource location, topic indexing) also creates a barrier for participants to learn about secure programming. 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.

5.4 The Need of Security Expertise Coordination

People join OSS community at a different age, with different backgrounds, different capacities and resources, and with different objectives. The issue of security expertise levels among OSS development members is critical, especially when considering a variety of domains of expertise ranging from strategic goal and problem-solving expertise to trained motor skills and operational expertise. Another problem is that secure programming is still not a well-known discipline in OSS communities, so there remains a lot of confusion as to what is secured code and what the project wants.

In our study, we observe that effective learning firstly results in coordinating necessary security expertise in the project, which enables a high level of security knowledge creation and the satisfaction of the learning process. In this study, expertise coordination is manifested through the two following 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. OSS communities can utilize security experts to define security requirements and best practices, help perform code reviews, and provides the necessary education for the software development staff [18]. Through this structural mechanism, the security knowledge is able to gain valuable insights from the organization to facilitate strategic decision making [20].

The term infostructure is commonly used to describe the infrastructure of information that is used in multiple disciplines. As indicated by Tilton [43], an infostructure is the layout of information in a manner such that it can be navigated – it is what's created any time an amount of information is organized in a useful fashion. 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 [35].

The role of expertise coordination is to provide access to the experience and knowledge, which help the learners reach their potentials. Software security knowledge can be abstracted, explicitly represented, codified, and accessed. In the OSS development, security expertise coordination not only facilitates a common understanding of security requirements but also specializes it in the context of the project development. It helps participants identify the location of the security information, and the most important for knowledge work, knowing where an answer to a problem.

5.5 Security Culture is the Key

Security culture reflects the belief and value of people that make up the organization. It is about actively practicing good security habits and making security-minded decisions [8, 29]. In short, security culture is the way our minds are programmed that will create different patterns of thinking, feeling, and actions for providing the security process [1]. It also includes all socio-cultural measures that support technical security methods in order for making security a natural aspect of organizational members' daily activities [38].

Culture shapes what a group defines as relevant knowledge, and this will directly affect which knowledge a unit focuses on [9]. Similarly, security culture decides how much security knowledge is disseminated within the community and what knowledge learners can learn. If an OSS project truly holds a value that software security is important, then particular behaviors and actions can be expected. As our study, the three projects, with different software domains and project stages, unfolding different security culture: Project A is at the young age and rapid production of function-based requirements was their strategy; Project B helps educate the community on writing secure code, and Project C focuses on proactive security auditing. 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.

6. LIMITATIONS

The study has some limitations. The observation was conducted in three OSS projects. It is reasonable to think that observation in several projects, covering more software types, could have given a more balanced result in the form of highlighting both hindrances and support for secure programming learning. Data collection and a major part of the analysis were conducted by the first author. More participation by different observers could have broadened the view of observations in the OSS communities.

7. CONCLUSIONS

This empirical study focuses on exploring learning about secure programming in open source software communities. Open source software has become a critical component and a key competency of the information and communication technology (ICT) ecosystems. While the number of found vulnerabilities in OSS is increasing, it is noteworthy that knowledge sharing and learning about secure programming in OSS communities has not gained much attention, and it is necessary to examine why knowledge-sharing mechanisms have not been effective despite efforts by OSS projects.

ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32

https://doi.org/10.1145/3178158.3178202

This study first addressed the learning opportunities of secure programming in OSS communities while investigating how mechanisms for sharing security knowledge have been implemented. Specifically, this study applied a socio-technical systems perspective, which systematically and holistically took into account the social context as well as technological aspects. Based on the socio-technical framework and context, we then examined the main factors that were once disproportionately considered in the learning process of secure programming in the context of OSS development and made suggestions for promoting a more effective as a central role for building robust software products.

In the context of distributed development, like OSS projects, learning is always to a great deal an individual exercise. People join OSS community at a different age, with different backgrounds, different capacities and resources, and with different objectives. The fields they came from are from any discipline that might lack formal, college-level software security training, they do not see an economic incentive for squeezing security thinking into their works and producing secure code. It is suggested that OSS communities have to establish rules and norms, roles and facilities, i.e., to offer opportunities for learning and self-development of secure programming knowledge for newcomers as well on the horizontal level between the experienced (but ever-learning) community members.

8. ACKNOWLEDGEMENT

The author would like to thank Professor Dr. Stewart Kowalski and Professor Dr. Rune Hjelsvold of Faculty of Information Technology and Electrical Engineering at Norwegian University of Science and Technology, who have made comments and suggestions in this paper.

9. REFERENCES

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

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

[3] Anderson, R. C., R. W. Kulhavy and T. Andre (1972). "Conditions under which feedback facilitates learning from programmed lessons." Journal of Educational Psychology. volume 63, issue 3, pages 186.

[4] Au, Y. A., D. Carpenter, X. Chen and J. G. Clark (2009). "Virtual organizational learning in open source software development projects." Information & Management. volume 46, issue 1, pages 9-15.

[5] 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.

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

[7] Cerone, A. and S. K. Sowe (2010). "Using free/libre open source software projects as e-learning tools." Electronic Communications of the EASST.

[8] Da Veiga, A. and J. H. Eloff (2010). "A framework and assessment instrument for information security culture." Computers & security. volume 29, issue 2, pages 196-207.

[9] David, W. and L. Fahey (2000). "Diagnosing cultural barriers to knowledge management." The Academy of management executive. volume 14, issue 4, pages 113-127.

[10] Ducheneaut, N. (2005). "Socialization in an open source software community: A socio-technical analysis." Computer Supported Cooperative Work (CSCW). volume 14, issue 4, pages 323-368.

[11] Feller, J., P. Finnegan, D. Kelly and M. MacNamara (2006). Developing open source software: a community-based analysis of research. Social Inclusion: Societal and Organizational Implications for Information Systems, Springer: 261-278.

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

[13] Fernandes, S., M. H. Martinho, A. Cerone and L. S. Barbosa (2013). "Integrating formal and informal learning through a FLOSS-based innovative approach". International Conference on Collaboration and Technology, Springer.

[14] Hardi, J. (2010). "Situated Learning among Open Source Software Developers: The Case of Google Chrome Project.".

[15] Hemetsberger, A. and C. Reinhardt (2004). "Sharing and creating knowledge in open-source communities: the case of KDE". Paper for Fifth European Conference on Organizational Knowledge, Learning, and Capabilities, Innsbruck.

[16] Hemetsberger, A. and C. Reinhardt (2006). "Learning and knowledge-building in open-source communities: A social-experiential approach." Management learning. volume 37, issue 2, pages 187-214.

[17] Hippel, E. v. and G. v. Krogh (2003). "Open source software and the “private-collective” innovation model: Issues for organization science." Organization science. volume 14, issue 2, pages 209-223.

[18] Howard, M. (2004). "Building more secure software with improved development processes." IEEE Security & Privacy. volume 2, issue 6, pages 63-65.

ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32

https://doi.org/10.1145/3178158.3178202

[19] 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.

[20] Kayworth, T. and D. Whitten (2012). "Effective information security requires a balance of social and technology factors." volume, issue, pages.

[21] Kogut, B. and A. Metiu (2001). "Open - source software development and distributed innovation." Oxford review of economic policy. volume 17, issue 2, pages 248-264.

[22] Kolb, D. (1984). "Experiential learning as the science of learning and development." Englewood Cliffs, NJ: Prentice Hall.

[23] 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.

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

[25] Lakhani, K. R. and E. Von Hippel (2003). "How open source software works:“free” user-to-user assistance." Research Policy. volume 32, issue 6, pages 923-943.

[26] Lave, J. (1991). "Situating learning in communities of practice." Perspectives on socially shared cognition. volume 2, issue, pages 63-82.

[27] Lave, J. and E. Wenger (1991). "Situated learning: Legitimate peripheral participation." Cambridge university press.

[28] Lu, S. C. and N. Jing (2009). "A socio-technical negotiation approach for collaborative design in software engineering." International Journal of Collaborative Engineering. volume 1, issue 1-2, pages 185-209.

[29] Martins, A. and J. Elofe (2002). Information security culture. Security in the information society, Springer: 203-214.

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

[31] Mayo, D. G. (1996). "Error and the growth of experimental knowledge." University of Chicago Press.

[32] Mirbel, I. (2009). "OFLOSSC, an Ontology for Supporting Open Source Development Communities". ICEIS (4).

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

[34] Oliveira, D., M. Rosenthal, N. Morin, K.-C. Yeh, J. Cappos and Y. Zhuang (2014). "It's the psychology stupid: how heuristics explain software vulnerabilities and how priming can illuminate developer's blind spots". Proceedings of the 30th Annual Computer Security Applications Conference, ACM.

[35] 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.

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

[37] Scacchi, W., J. Feller, B. Fitzgerald, S. Hissam and K. Lakhani (2006). "Understanding free/open source software development processes." Software Process: Improvement and Practice. volume 11, issue 2, pages 95-105.

[38] Schlienger, T. and S. Teufel (2003). "Analyzing information security culture: increased trust by an appropriate information security culture". Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on, IEEE.

[39] Seifert, C. M. and E. L. Hutchins (1988). "Learning from error." AMERICAN SOCIETY FOR ENGINEERING EDUCATION WASHINGTON DC.

[40] 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.

[41] Singh, V. and L. Holt (2013). "Learning and best practices for learning in open-source software communities." Computers & Education. volume 63, issue, pages 98-108.

[42] Sowe, S. K., A. Karoulis and I. Stamelos (2006). A constructivist view of knowledge management in open source virtual communities. Managing learning in virtual settings: the role of context, IGI Global: 290-308.

[43] Tilton, J. (1994). "What is an Infostructure." Web page: http://library.creatifica.com/information-architecture/infostructure-coining-the-term.pdf

[44] Tyre, M. J. and E. Von Hippel (1997). "The situated nature of adaptive learning in organizations." Organization science. volume 8, issue 1, pages 71-83.

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

[46] Wenger, E. (2000). "Communities of practice and social learning systems." Organization. volume 7, issue 2, pages 225-246.

ICIET '18: Proceedings of the 6th International Conference on Information and Education Technology January 2018, Pages 25–32

https://doi.org/10.1145/3178158.3178202

[47] Ye, Y., Y. Yamamoto and K. Nakakoji (2007). "A socio-technical framework for supporting programmers". Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ACM.