Hyper Contextual Software Security Management for Open Source Software

Shao-Fang Wen

Norwegian Information Security Lab
Faculty of Computer Science and Media Technology
Norwegian University of Science and Technology, Norway

shao-fang.wen@ntnu.no

Abstract. Since the turn of the century, open source software (OSS) has been an active and dynamic research area. OSS development and maintenance are highly distributed processes that involve a multitude of supporting tools and resources. OSS communities use numerous knowledge sources while working on a certain task to help them secure the software products. These not only include security incidents statistics and best practice documents that are published in the open literatures or online communities, but also social networking tools. This often results in additional challenges, as not every OSS project member can correlate particular learned security information with their working context. This position paper outlines the security problems in OSS and describes the use of socio-technical system theory and ontology technologies to capture and model software security knowledge. Our research aims to develop and test a hyper-contextual, knowledge-based environment that stores and process security knowledge to facilitate retrieval in context, and thus allows the non-linearly correlated knowledge between contexts to be identified and transferred between and among OSS developers and users.

1. Introduction

Open source software (OSS) has become increasingly important and has attracted developers from both public and private sectors. Open source model, as a radically new software development model, begins in the mid-90s. Since then, a good deal of software created by open source model have been widely adopted and used by various industries. The 2015 Future of Open Source Survey [11] reported that, 78% of companies run operations on open source, and 55% of respondents said open source delivers superior security [6]. This reputation can be contributed to community development model in OSS development and the resulting purview by the "many eyes" of developers worldwide. Yet, of the 8,000-13,000 vulnerabilities detected annually, about 40% impact open source software [10]. These vulnerabilities open some of the most critical OSS

projects to potential exploit: Heartbleed and Logjam (in OpenSSL); Shellshock (in bash); Venom (in QEMU and OSS hypervisors), and NetUSB (in the Linux kernel). While both the quantity and severity of vulnerabilities are increasing in OSS, its development and maintenance present a unique management and software security challenges.

OSS development and maintenance take place in a distributed environment, involving a multitude of supporting tools and resources, integrated in complex and often partially defined workflows and processes [15]. OSS communities use numerous knowledge sources while working on a certain task to help them secure the software products. These not only include security incidents statistics and best practices documents published in the open literatures or online communities, such as Open Web Application Security Project[1] (OWASP), Build Security In[2] (BSI) project and Open Sourced Vulnerability Database[3] (OSVDB), but also social networking tools, such as group mails, dynamic blogs and wiki systems. Software engineers have these security resources at their disposal, but this also results in a form of information overload. They have difficulties correlating particular learned vulnerabilities or security information with their working context [12]. Identifying security knowledge that are applicable in a given context can become a major challenge for OSS. The implicit knowledge is often lost, since it is not captured by today's security management environments. A similar security case might have been successfully resolved by a different developer using a solution that other members are unaware of, but if this knowledge is not captured, stored, and delivered in a context-sensitive manner to the team even the whole community, it cannot serve as a "cognitive map" for future problem-solving. If we can capture, combine and apply this 'ambient' (contextual) knowledge into coherent chunks, priming it when software engineers need it, we can bring OSS security to a completely new level: a hyper-contextual, active software security management environment that can provide a collective memory for the communication within the communities. We propose the notion of Hyper Contextual Software Security Management that stores and processes security knowledge to facilitate retrieval in context, and thus allows the non-linearly correlated knowledge between contexts to be identified and transferred between OSS developers and users.

On the conceptual side, this research is based on socio-technical system theory and ontology technologies. Software engineering is a multifaceted domain, which stretches from low-level technical aspects (e.g., source code, operating systems and tools such as compilers and editors) to organizational and legal concerns (e.g., prescribed process,

[1] Open Web Application Security Project (OWASP) is an online community which creates freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security. https://www.owasp.org/

[2] Build Security In (BSI) is a collaborative effort that provides practices, tools, guidelines, principles, and other resources that software developers, architects, and security practitioners can use to build security into software in every phase of its development. https://buildsecurityin.us-cert.gov/

[3] Open Sourced Vulnerability Database (OSVDB) is an independent and open-sourced database which aims to provide accurate, detailed, and unbiased technical information on security vulnerabilities. https://blog.osvdb.org/

international standards) to social and cognitive aspects (e.g., communication behavior and cognitive models) [5]. Providing adaptive support that addresses the security concerns is difficult, due to the different representations and interrelationships that exist in the context of software engineering and knowledge resources. As Scacchi [13] points out, the meaning of open source in the socio-technical context is broader than its technical definition, and includes communities of practice, social practices, technical cultures, and uses. In this research, we will apply a socio-technical systems perspective to address the security characteristics in open source phenomenon. Based on the observed socio-technical context, we examine the main factors that were once disproportionately considered in software security knowledge. Ontology is then used in knowledge modeling since it's a good approach to systematically categorize various concepts and describe their relationships [3]. By capturing knowledge from various perspectives through ontology population, we can build an extensible, distributed security knowledge base.

Our approach lies in explicitly describing and abstracting the security knowledge that are needed by OSS developers and also other stakeholders in the communities to successfully perform their particular tasks. This extensible knowledge model not only includes existing security standards and guidelines, but also their relevance within a certain development or maintenance context by using knowledge collection through investigating the behavior of team members while solving similar security events. This research strives not to propose the adaptation of a new tool or development process, but rather examine how existing resources can be integrated to implement the next generation of software security management environments, which is an important contribution neglected by current researches.

2. Research Goal and Research Questions

The goal of this research work is to develop and test a hyper-contextual, knowledge –based system that would facilitate open source communities to effectively offer appropriate secured software products. We seek to examine the hypothesis: *Hyper Contextual Software Security Management can improve security quality of software product that are developed, delivered and maintained by open source communities.*

To better understand the scope and magnitude of the research goal, four research studies along with their respective research questions were formulated as follows:

Study-1: Research question 1

At first it is important to identify and establish the magnitude of the real-world situation, including current practices (tools, knowledge and other resources) used in OSS communities. When responding to these requirements, research question 1 is split into two sub-research questions:

RQ1 (a): What are the current issues and challenges facing secure software products that are developed, delivered and maintained by open source communities?

RQ1 (b): What are the strengths and weaknesses, technical and non-technical, of software security practices used by open source communities?

Study-2: Research question 2

After study–1, it is imperative to investigate, identify, and develop software security knowledge, technical and non-technical, that could appropriately be integrated into the security knowledge management system. Therefore, the second research question is formulated as:

RQ2: What contemporary software security information can be contextualized and formalized into the proposed software security knowledge management system for securing software products in OSS communities?

Study-3: Research question 3

After study-2, which identifies the security characteristics and factors that are appropriate, it is necessary to develop a system for formalizing and integrating this security information into an ontological knowledge model. Therefore, the third research question is formulated as:

RQ3: How can proposed software security information, technical and non-technical, be contextualized and formalized into an integrated ontological model to form a knowledge management system for securing software products in OSS communities?

Study-4: Research question 4

After study–3, it is necessary to evaluate the proposed system addresses in RQ 3, in the studied environment. Therefore, the fourth research question is formulated as:

RQ4: How can the proposed knowledge management system be evaluated to effectively meet the demands for securing software products in OSS communities?

3. The Socio-Technical Framework

In this research, we will make an interpretive inquiry in the context of OSS evaluation using a socio-technical framework provided by Stewart Kowalski [9]. The socio-technical framework contains two basic models: a dynamic model of socio-technical changes, called the *socio-technical system* (see Figure 3-1), and a static one, called the *security-by-consensus* (SBC) *model* or stack (see Figure 3-2). At the abstract level, the socio-technical system is divided into two subsystems, social and technical. Within a given sub-system there are further sub-systems. The former (social) has culture and structures, and the latter (technical) has methods and machines. From the system theory/s point of view, inter-dependencies between system levels make a system adjust for attaining equilibrium. The process is referred to as homeostasis state. For instance, if new hardware is introduced into one of the technical sub-systems, for instance, the machine sub-system; the whole system will strive to achieve homeostasis. This suggests that changes in one sub-system may cause disturbances in other sub-systems and consequently to the entire system.

Reflecting the static nature of the socio-technical systems, the SBC stack is a multi-level structure that divides security measures into hierarchical levels of control. The

social sub-system include following security measures: ethical and cultural norms, legal and contractual documents, administrational and managerial policies, and operational and procedural guidelines. Similarly, the technical sub-system consists mechanical and electronic, hardware, operating systems, application systems, and data. Other aspects are: store, process, collect, and communication.

In the socio-technical framework, each system interacts with other systems rather than being an isolated system. Internal and external changes—both social and technical—will affect system security. Therefore, systematic deployment of security measures is required. In particular, this framework has been applied to evaluate threat modeling in software supply chain [1], business process re-engineering [4], and an information security maturity model [7]. The application of the socio-technical framework to software analysis is an appropriate and legitimate way of understanding the intrinsic context in open source phenomenon. It provides a way to perform system analysis through a systemic–holistic perspective [8].

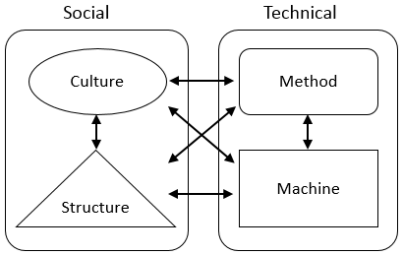

Figure 3-1: Socio-technical model
(Kowalski [9], page 10)

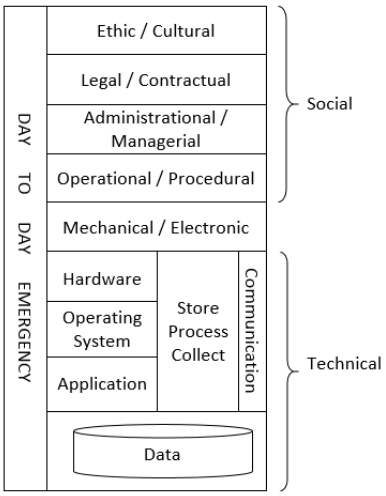

Figure 3-2: The SBC model
(Kowalski [9], page19)

4. Methodology

Since the main goal of this research is to produce an artifact, the design science appears as an appropriate methodology of our research. Design science research (DSR) methodology can be conducted when creating innovations and ideas that define technical capabilities and products through which the development process of artifacts can be effectively and efficiently accomplished [2, 14]. The design science approach applied for this study is based on work presented by Vaishnavi and Kuechler [14]. Figure 4-1 represents design science research process model.

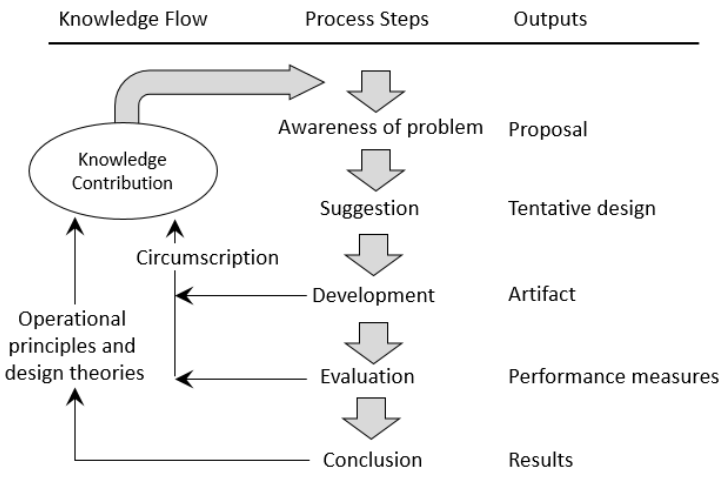

Figure 4-1: Design science research process model
(Vaishnavi and Kuechler [14])

Consequently, research studies and their corresponding research questions that are linked to the DSR process (steps), are connected to research activities as follows:

- Study–1, RQ 1 (a) (b) links to awareness of the real-world problem step in DSR. Research activities involves conducting extensive literature review and case-studies (questionnaires) in the selected OSS communities on various issues related to software security management for OSS, as well as issues related to security knowledge learning in the real-world environment.
- Study–2, RQ 2 links to suggestions for a tentative design step in DSR. The study involved conducting an extensive literature review on various ontological models, security standards, and best practices. Questionnaires will be prepared, aimed at gathering OSS stakeholders' views on proposed security knowledge model and their respective security requirements control areas, technical and non-technical.
- Study–3, RQ 3 links to developing the artifact step in DSR. This study will apply action research strategy to continuously build the artifact and improve its quality in the context of focused efforts. We repeat the process as a spiral of cycles of action and research in four main phases: planning, acting, observing and reflecting.
- Study–4, RQ 4 links to evaluating the proposed system step in DSR. This study will be conducted in the selected OSS communities. Evaluation methods could be either practical, theoretical or both. Practical evaluation methods include tests, experiments and analysis. Theoretical evaluation methods include observations and descriptions. They involve use of qualitative techniques such as case-study, field-study, informed argumentation and scenario analyses.

Table 4-1 summarizes the presentation of this section. In the table, the linkage between research studies, research questions, and corresponding DSR steps and research activities is given.

Table 4-1: Summary of the research studies, questions and corresponding DSR steps and research activities

Research Study	**Research Question**	**DSR Step**	**Research Activity**
Study-1	RQ1 (a) (b)	Awareness of the real-world problem	Literature review, questionnaires, physical observation, interview
Study-2	RQ2	Suggestion for tentative design	Literature review, questionnaires, physical observation, interview
Study-3	RQ3	Developing the artifact	Action(development), observation, reflection
Study-4	RQ4	Evaluating the artifact	Practical evaluation: testing, experimental and analytical Theoretical evaluation: case-study, field-study, and scenario analyses

5. Conclusions

Given the increased complexity and importance of open source software in today's society and an increased knowledge gap between security information available and security information used and practiced, our position is that new socio-technical tools and approaches are needed. My position is that hyper contextual software security management is a means to help fill this gap by bringing the ability to place information in an appropriate context and to use knowledge in an ever changing global security environment.

Acknowledgement

The author would like to thank Professor Dr. Stewart Kowalski and Professor Dr. Rune Hjelsvold of Faculty of Computer Science and Media Technology at Norwegian University of Science and Technology, who have made contributions to the ideas described in this paper.

Reference

[1] 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.

[2] Alan, R. H., S. T. March, J. Park and S. Ram (2004). "Design science in information systems research." MIS quarterly. volume 28, issue 1, pages 75-105.

[3] Bäck, A., S. Vainikainen, C. Södergård and H. Juhola (2003). "Semantic Web Technologies in Knowledge Management". ELPUB.

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

[5] Brooks, R. (1983). "Towards a theory of the comprehension of computer programs." International journal of man-machine studies. volume 18, issue 6, pages 543-554.

[6] Hoepman, J.-H. and B. Jacobs (2007). "Increased security through open source." Communications of the ACM. volume 50, issue 1, pages 79-83.

[7] Karokola, G., S. Kowalski and L. Yngström (2011). "Towards An Information Security Maturity Model for Secure e-Government Services: A Stakeholders View". HAISA.

[8] Karokola, G. R., S. Kowalski, G. J. Mwakalinga and V. Rukiza (2011). "Secure e-Government Adoption: A Case Study of Tanzania". European Security Conference.

[9] 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.

[10] Martin, B., C. Sullo and J. Kouns. (2015). "OSVDB: open source vulnerability database." Electronic document. https://blog.osvdb.org/category/vulnerability-statistics/.

[11] NorthBridge (2015). "2015 Future of Open Source Survey." Electronic document. http://www.northbridge.com/open-source

[12] 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.

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

[14] Vaishnavi, V. and W. Kuechler (2004). "Design research in information systems." Electronic document. http://desrist.org/design-research-in-information-systems/.

[15] Von KROGh, G. and S. Spaeth (2007). "The open source software phenomenon: Characteristics that promote research." The Journal of Strategic Information Systems. volume 16, issue 3, pages 236-253.

