education sciences

MDPI

Article

Context-Based Support to Enhance Developers' Learning of Software Security

Shao-Fang Wen

Department of Information Security and Communication Technology, Norwegian University of Science and Technology, 2815 Gjøvik, Norway; shao-fang.wen@ntnu.no

Abstract: Software security is an ongoing problem, largely due to a lack of security knowledge among software developers from diverse backgrounds. To counter this, security experts are attempting to offer a broad range of knowledge resources to enlighten developers about increasing cybersecurity threats. Unfortunately, the abundance of knowledge resources does not seem to have much of an impact on reducing the issue of software security. The ineffective teaching and learning approaches for software security have created difficulties for developers in learning security knowledge. This research employs a four-cycle of Design Science Research Methodology (DSRM) to integrate necessary elements in the development of a context-based learning system for security education and learning. The final artifact is an ontology-based web application that facilitates a contextualized learning process by providing security knowledge through contextual software cases. Through evaluation in pedagogical and software development environments, it is proven to contribute a viable solution to the problem domain. While these results are positive, the innovative context-based artifact benefits not only the domain of software engineering but also other educational fields, such as information security and computer security.

Keywords: software security; security education; knowledge management; context-based; design science

Citation: Wen, S.-F. Context-Based Support to Enhance Developers' Learning of Software Security. *Educ. Sci.* **2023**, *13*, 631. https://doi.org/10.3390/educsci13070631

Academic Editors: Katerina Tzafilkou, Anastasios A. Economides and Maria Perifanou

Received: 19 April 2023
Revised: 8 June 2023
Accepted: 16 June 2023
Published: 21 June 2023

1. Introduction

As the digital space grows ever more impactful, there has never been a better time to become a software developer. A Stack Overflow survey [1] reported that of over 80,000 people surveyed, nearly 60% had acquired coding skills through online resources. Younger respondents tend to learn from online courses, forums, and other online resources. Older respondents, on the other hand, learned from more traditional mediums like school and books. This demonstrates the versatility and opportunities present in software development and presents an encouraging landscape for newcomers: learning opportunities are plentiful and accessible. Despite a large number of self-taught developers, many lack important fundamentals when it comes to secure software and computer operations. This deficiency is also present in curricula of computer science and software engineering, which struggle to equip students with essential knowledge of software security. Veracode and DevOps.com research [2] determined that a surprisingly small percentage of undergraduate computer science programs require security coursework (2.8%) and just 24% of college-educated developers reported taking such courses as part of their education [3]. Without fundamental security knowledge (the terms "security knowledge," "secure software knowledge," and "software security knowledge" are used as inclusive terms in this thesis; they all refer to knowledge of engineering software that allows one to ensure that software continues to function correctly under malicious attacks), developers can still set up their software as intended; however, they may not have the capacity to detect issues when coding malfunctions or recognize vulnerabilities upon discovery. Consequently, late identification of security weaknesses can result in delayed product launches, last-minute fixes, and potentially insecure applications entering the market [4].

For developers to stay informed on the latest security trends and developments, security communities and industries have made an abundance of learning materials available, ranging from checklists to best practices. Sources that developers can use to access these materials include books, online resources (videos, blogs, or forums), or even scientific databases. However, being able to understand this content requires having a strong foundation in basic security education: without this understanding, it can be difficult for developers to identify the relevant information among countless online resources. Over the past few years, there has been exponential growth in learning resources, creating a surplus of information. This puts learners under an excessive cognitive burden that hinders their ability to learn quickly and easily from multiple sources [5,6]. Research suggests that learners' attitudes toward learning tend to diminish over the course of a session, due to an overloaded mental state [7]. Currently, available security learning materials are satisfactory in terms of structure yet somewhat limited in scope and inflexible in meeting individual preferences. Conventional teaching approaches in the field of software security often focus on specific security topics, following a structured and logical order that begins with abstract security concepts and theories. Without contextualizing the knowledge, learners may struggle to connect it with their existing programming knowledge, making it difficult for them to apply the necessary skills when faced with real-world security issues or scenarios. Consequently, they might consider secure software development too difficult to attempt [8,9].

Researchers have suggested that learners require motivation and access to appropriate teaching facilities for effective learning to take place [10]. To address these needs, an engaged learning environment can be developed. Doing so fosters intrinsic motivation in students and increases the chances that teaching will be successful. According to Jonassen and Land [11], "Learners must be introduced to the context of the problem and its relevance, and this must be done in a way that motivates and engages them". Context and the particulars of that context can provide a powerful motivation for learning [12]. Numerous studies have indicated that studying from a context and then abstracting the knowledge gained to be able to use it in a new context is a common way of learning programming that has been observed extensively in both new and experienced programmers [9,13].

To address the security challenges in software development environments, this research proposes a context-based learning approach. We developed an online learning platform in which security knowledge can be contextualized in a way that makes sense to programmers, thus enhancing their understanding. We hypothesize that through the integration of context-based learning methods, the application of this learning tool is anticipated to enhance software developers' learning experience in security.

To achieve the research objectives, three main research questions (RQs) are formulated to guide the research activities.

RQ 1: How can context-based approaches be applied in software security to motivate learners and improve learning outcomes?

While traditional security instruction design struggles to foster successful learning outcomes, context-based teaching and learning have demonstrated promise in a variety of scientific learning environments. This research proposes to investigate the effectiveness of a context-based approach for software security learning, to understand the ability it has to motivate students. We want to understand how currently available concepts can be developed and applied to effectively teach and learn software security knowledge.

RQ 2: What approach can be taken to develop an ontology that effectively manages contextualized knowledge related to software security?

This research has been designed to assist with knowledge management in security learning by remodeling security knowledge so that it can be retrieved in consideration of real-world cases. Ontologies provide an ideal foundation for this project, as they enable the capture and construction of domain knowledge, along with the representation of a skeletal framework [14]. To answer our research question, we will first address the design pattern of an ontology to manage contextualized and theoretical security knowledge. We then

intend to apply ontology evaluation techniques to assess the ontological artifact's feasibility and applicability in constructing an ontology-based learning system.

RQ 3: What strategies can be employed to build a learning system for software security that incorporates contextualized content, and what impact does this system have on the overall learning outcomes?

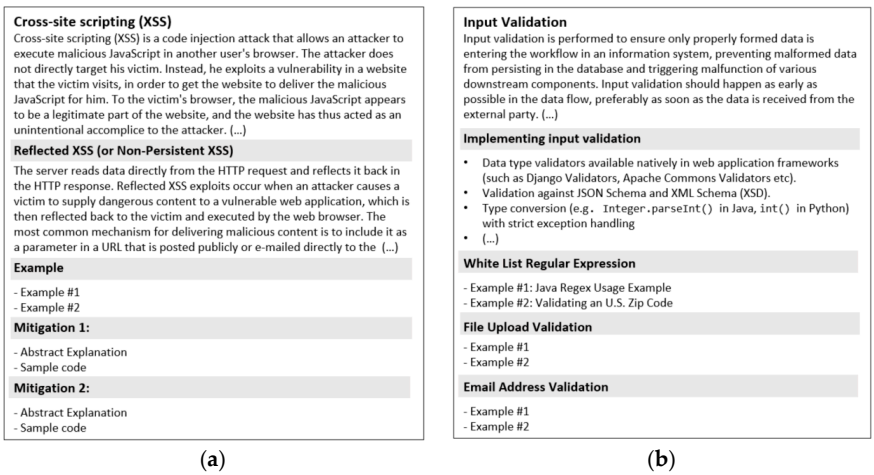
RQ 2 and RQ 3 examine the proposed context-based learning approach and ontological knowledge base, respectively, while RQ 4 details how to develop a web-based software security learning system that utilizes the developed ontology as its foundation. To bring this proof-of-concept system to a more generalized proof-of-use assessment, an implementation must be completed in various situations, with consideration of whether or not it solves the problem at hand.

The rest of this paper is organized as follows. After the introduction, in Section 2, we introduce the scientific background and relative work. Section 3 describes the overall design consideration of the proposed learning artifact. Section 4 presents the research methodology and the research design, followed by a discussion in Section 5. We describe the threats to the validity of this research work in Section 6. Lastly, Section 7 presents the conclusions and future research opportunities.

2. Scientific Background and Related Work

2.1. Convention Software Security Teaching and Learning

In the learning process, learning materials are one of the main factors to be considered by the instructor because they can contribute to the acceptance of students of the knowledge presented. Learning material can consist of various forms and formats depending on the teaching methods. When most institutions plan and develop security learning materials, either textbooks, lectures, or online courses, they commonly use conventional approaches to guide the process. Such conventional learning materials and approaches are commonly made up of two distinct methods: black-hat concepts and white-hat concepts (offense/defense, construction/destruction) [15]. Figure 1 illustrates the two types of learning materials. The black hat/white hat concepts apply the classic western "bad guy/good guy" concept to software security. A black hat refers to a hacker who tries to break into a system with malicious intent. Black-hat actions include destructive activities such as attacks and exploits [15]. Using a black-hat approach in software security implies thinking proactively about ways that a system could be exploited. A white hat refers to an individual who identifies a vulnerability in a system and reports it to the system owners. White-hat actions include constructive activities such as design, defense, and functionality [15]. Using a white-hat approach in software development includes building defense into a system, often using information from a black-hat history.



(**a**) (**b**)

Figure 1. Two types of conventional learning materials for software security: (**a**) the black-hat approach, and (**b**) the white-hat approach.

The conventional learning materials typically address particular security topics, and the starting point of instructions consists of basic security concepts and theories, which are taught in a logical order and structure. Moreover, these learning materials are often written in the form of a reference manual or a guide to a particular security certification, which is more effective in training security experts. However, it is difficult for developers to correlate what they are learning to their programming experience and, further, to link the security knowledge to real software scenarios. In this approach, the interests and thoughts of developers and the knowledge they already possess are not taken into account, which could lead to forced concept development and misconceptions. An ideal learning process should therefore also be guided by the motives, skills, and pre-knowledge of students. Since security learners have to demonstrate the applicability of the knowledge through experience to understand their practical use [15], the learning materials presented must provide meaning for learners, allowing them to learn security principles close to real-world situations that are of particular interest to them.

2.2. Context-Based Teaching and Learning

According to Dey [16], "context is a set of information used to characterize a situation of an entity". In the real world, context is a complex description of the knowledge shared in physical, historical, and other circumstances where actions or events happen [12]. Brézillon [17] points out that knowledge comes from a variety of contexts that cannot be accurately understood without context. The context can provide a major meaning to knowledge, promoting a more effective comprehension of a determined situation in collaborative work [18]. In field observations of the usage of an organizational knowledge management system that stores knowledge about UNIX problems, Ackerman [19] found that users chose not to use the solution provided by the system because they could not determine the appropriateness of the solution without knowing the context in which the solution has been applied, such as the size of the UNIX installation and the organizational setting. Addressing this shortcoming requires knowledge built around real-world scenarios that actively engage learners [8,20].

Context can increase the information content of natural language utterances and facilitate learning [21,22]. Psychology and education researchers have demonstrated that when knowledge is learned in a context similar to that in which the skills will be needed, the application of the learning to the new context may be more likely [16,23]. Predmore [24] showed that learning about knowledge content through real-world experience is important because "once [students] can see the real-world relevance of what they're learning, they become interested and motivated." The book *How People Learn* [25] also pointed out that motivation is critical for learning, enabling knowledge transfer to occur. If students do not learn the material well in the first place, they cannot possibly transfer it to new situations. As stated in the book "Learners of all ages are more motivated when they can see the usefulness of what they are learning and when they can use that information to do something that has an impact on others".

Context-based learning approaches aim to bring science learning closer to the lives and interests of learners and to illustrate how using familiar contexts can increase their interest in science and therefore enhance their understanding [26]. Researchers have identified several interrelated problems and challenges in science education and learning that context-based learning approaches intend to address: (a) curricula are overloaded [27,28]; (b) too many isolated facts and concepts prevent students from developing a worthwhile "mental model" [29], and (c) an excessive emphasis on correct explanations and solid foundations leaves students confused about reasons for learning science [27,30]. Given the ongoing challenges in security education, context-based learning appears to hold promise in addressing the specific needs of software security education. This approach is not new, and education researchers have emphasized learning in context over the years; however, such approaches are not embraced in practice in the domain of software security, and much remains to be learned about designing learning support artifacts for use in context-based education.

2.3. Tool-Based Support for Learning Software Security

Some efforts have been made to enhance software security education and learning using tool-based learning approaches. In this section, various types of security education tools from the literature are briefly introduced.

Atsuo Hazeyama et al. [31,32] proposed an artifact-driven learning process for software security as well as an online learning environment utilizing a body of knowledge for security education. In the learning process, learners conduct secure software development by inputting artifacts that were created in a traditional software engineering course, such as requirements specification, use case diagram, and test specification. The learning flow considers security after considering the functional requirements of a system. The designed learning environment provides functionalities for maintaining artifacts that are inputs for learning about software security and outputs from that learning, furthermore giving relationships between artifacts and reference information. Learners proceed to learn about software security by referring to the available information.

In addition to online programs, the Integrated Development Environment (IDE) plug-in has been applied to teaching students or programmers about security awareness. The University of North Carolina at Charlotte (UNCC) has designed and developed an Educational Security in the Integrated Development Environment (ESIDE) plug-in for Eclipse (Eclipse is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment: https://www.eclipse.org, accessed on 2 November 2022) that delivers real-time secure programming instructional support as students write code [33,34], similar to the underline in a word processing spell checker. The tool is designed to improve student awareness and understanding of security vulnerabilities and to increase the utilization of secure programming techniques in assignments. ESIDE aims to provide educational interventions for more advanced students (a senior and masters-level web development course) [33], and the tool only works on Eclipse IDE for Java and cannot support other platforms, for example the Android IDE.

Visualization is another approach in teaching software security courses, which heavily uses images, diagrams, or animations to communicate messages [35]. To integrate visualization techniques into classroom instruction, Yuan [36] developed three visualization and animation tools that demonstrate various information security concepts. The information security concepts illustrated include packet sniffer and related computer network concepts, the Kerberos authentication architecture, and wireless network attacks, through the usage of Macromedia's Flash software. Bishop et al. [37] have developed a Concept Map (http://spc.cs.ucdavis.edu/index.php/conceptmap, accessed on 2 November 2022) of secure programming to visualize the relevant body of knowledge, which assists students in understanding complex concepts, principles, and ideas and the important relationships between them. Their concept maps are assessments designed to identify students' misconceptions; the questions, scoring procedures, and interpretations are consistent and in adherence with a predetermined standard. The results from the concept map are primarily intended to improve pedagogy, though the results can be used to help instructors make comparisons of teaching over time.

Furthermore, video games (game-based learning), such as CyberCIEGE [38] and hACMEgane [39], are approaches taken to stem the declining interest and enrollment in computing courses, where students explore relevant security aspects of games in a learning context designed by the instructor. CyberCIEGE (https://my.nps.edu/web/c3o/cyberciege, accessed on 2 November 2022) is a free tool that can be downloaded from the Internet for that purpose. Students can build their networking environment virtually and learn the possible threats that affect their network based on their design. Through available security scenarios, students will learn security through the consequences of their choice while they build their network. In hACMEgame, games are organized as a series of levels where the player must overcome a set of challenges to unlock access to the next level. Each level focuses on a set of well-known security vulnerabilities.

The web-based security learning tool proposed in this paper differentiates from the previous works in two main aspects:

(1) The tool is context-based, in which context-based learning is facilitated in the learning process.
(2) The tool is ontology-based, in which the security knowledge is modeled with contextual situations and incorporates theoretical knowledge to complement the concrete description.

3. Design Consideration

The basic concept of the contextualized security learning system is to facilitate the contextual learning process by providing contextualized access to security knowledge through real software application scenarios. Figure 2 depicts the design consideration of the proposed contextualized learning system for software security. To develop this kind of learning system, we first proposed a context-based learning approach as the main pedagogy. This will foster a deeper understanding of security through established contexts. To collect and utilize context-sensitive security information, it is necessary to embrace a wide range of knowledge while maintaining comprehendible formatting. As such, building a comprehensive knowledge base is paramount. To address these needs, an ontology-based knowledge model has been implemented, as it facilitates the capture of domain knowledge and provides an integrated hub with which to represent the core data [14]. The proposed learning approach regulates contextualized learning process about software security, outlines the requirements for developing such user interfaces for knowledge presentation, and provides design guidance for security knowledge modeling.

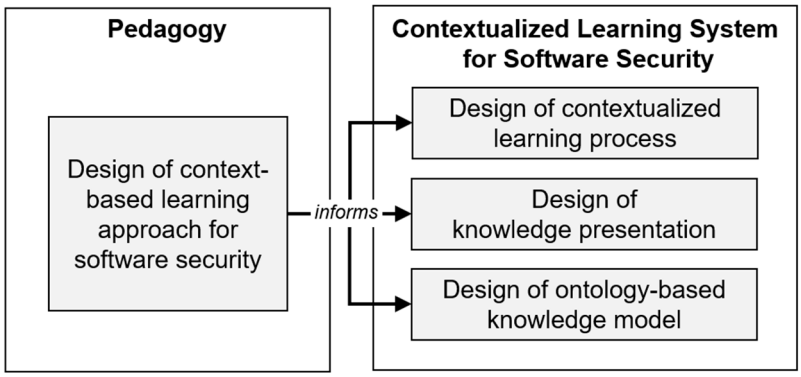


Figure 2. Design consideration of the proposed learning artifact.

4. Research Methodology

The development of the proposed learning system falls within the Design Science Research (DSR) methodology genre, which focuses on the development of an applicable and practical artifact, rather than on the creation of a design theory [40]. The study objective of DSR is the creation and use of artifacts that can advance individual as well as organizational and societal flourishing. DSR combines applied design with the generation of theoretical knowledge in the pursuit of problem-solving. It tackles real problems that rarely have optimal solutions and instead defines and pursues goals that provide satisfactory solutions [41]. Thus, based on the research objectives and considering the practical tasks when designing artifacts, DSR was adopted as the main research methodology in this research.

For communication and to provide a comprehensible level of rigor in the design description, we followed the Design Science Research Methodology Process Model (DSRM process model to conduct this project, depicted in Figure 3) provided by Peffers et al. [42], which outlines pragmatic disciplines of the main considerations for successfully conducting DSR [40] and, most importantly, provides a clear and flexible process for conducting rigorous design science research. The activities of the DSRM process model are (1) the identification of problems and motivation, (2) the definition of the objectives of a solution,

(3) design and development, (4) demonstration, (5) evaluation, and (6) communication. In a DSR project, the research process frequently iterates between the development and evaluation phases rather than flowing in a waterfall fashion from one phase to the next [43]. Hevner et al. [44] (p. 89) term this iteration the "generate/test cycle."

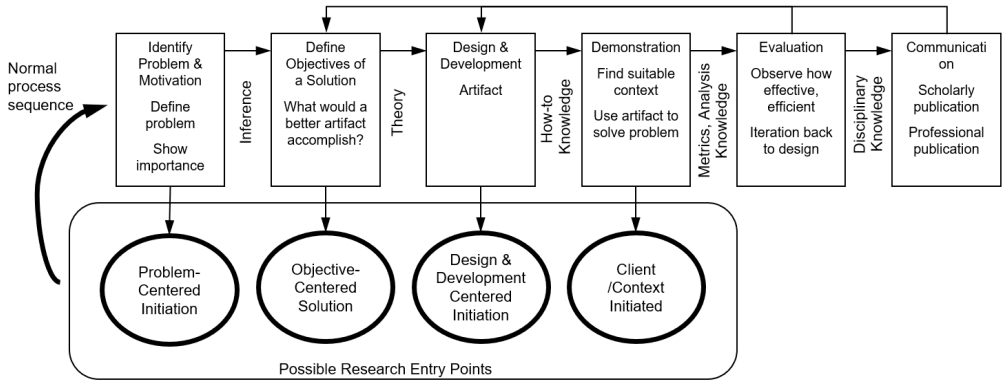


Figure 3. DSRM process model [42].

The DSRM process model was implemented in our research with a multi-iteration approach to ensure validity within a naturalistic setting involving potential users and domain experts. These iterations constituted individual design cycles (Figure 4) that allowed us to improve the artifact accordingly. With the given research problem and triggered motivation (addressed in Section 1), a four-design-cycle activity was carried out, in which each design cycle (DC) contained the following steps: objectives for a solution, design and development, demonstration, and evaluation. Rather than one evaluation at the end of the whole design process, in this research project, the artifact generated in each design cycle was put through a dedicated evaluation process to concretely testify to its validity. Further, each design cycle derived not only a designed artifact but also knowledge contributions consisting of communication and formalized learning.

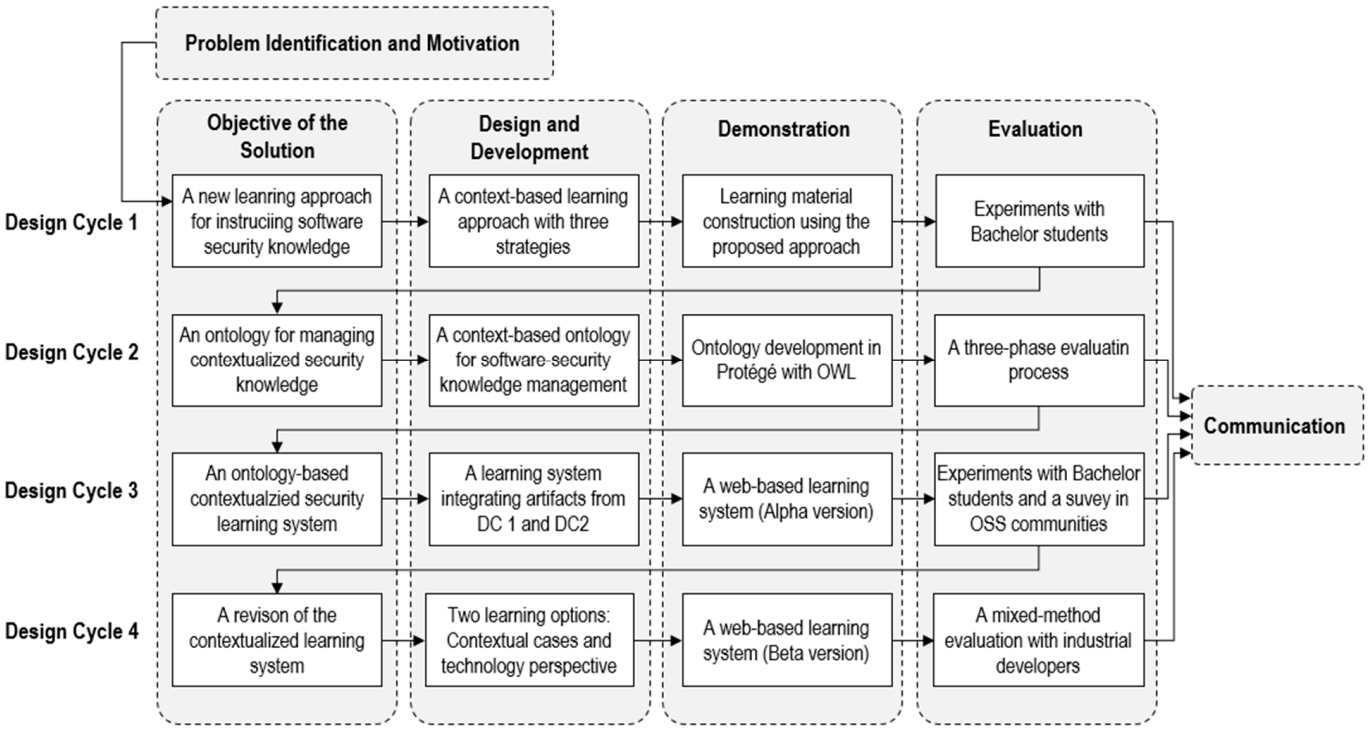


Figure 4. Iterations of DSR design cycles in this research.

4.1. DC1: A Context-Based Learning Approach for Software Security

4.1.1. Objective and Solution

The primary objective of the initial design cycle was to propose and evaluate a teaching and learning approach that serves as a suitable artifact in the instruction of software security knowledge. To this end, a context-based learning approach was first proposed, adopted from concepts of context-based learning and literature from psychology and education.

4.1.2. Design and Development

The proposed context-based approach includes three main strategies, illustrated in Figure 5. First, contextualized learning often takes the form of real-world examples or problem situations that are meaningful to the learners personally [45]. In the case of security knowledge, we suggest initiating learning by introducing a meaningful contextual framework that orients the student to software programming to effectively develop their security understanding. The application context design is intended to encourage drawing on learners' pre-existing software programming knowledge and anchor their security knowledge.

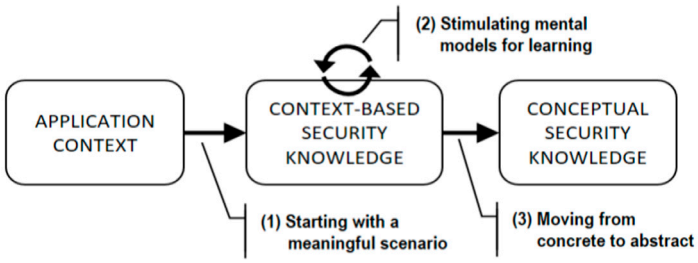


Figure 5. The proposed context-based approach for software security.

The second strategy is to structure the knowledge in a manner that encourages the development of mental models and improves the efficiency of learning. This strategy aims to allow learners to better access pertinent vulnerabilities, risks, and preventative measures in the required context. Generally, we encourage learners to answer three questions for each contextual scenario. These include (1) What types of attacks are prominent? (2) Why does this context have such attacks? And (3) How can these attacks be managed or avoided? Through exploring what–why–how questions, relationships between security concepts become clear, thereby aiding in the development of a more comprehensive mental model.

The third strategy to use in learning is guiding learners from context to abstraction. Abstract knowledge is based on underlying principles and is often derived from repeated contextual learning and problem-solving, which helps to cultivate skills that can be transferred to new situations. When concrete knowledge is associated with abstract information, learners can observe the relationship between practical and abstract concepts in tangible applications. For knowledge transfer to be effective, both the concrete aspects and the underlying abstraction must be fully understood by students. In this way, results can be seamlessly transferred from one context to another [46].

4.1.3. Demonstration and Evaluation

To prove the efficacy of our new context-based security learning method, we employ these three strategies to design and build materials concerning Web Application Security, with two particular vulnerabilities chosen—Cross-Site Scripting and SQL Injection. Figure 6 shows a simplified version of the learning material created for SQL injection; this was achieved using our proposed learning approach. The artifact was further evaluated to prove its effectiveness in improving learners' learning outcomes in studying software security. The evaluation method adopted in this design cycle is a controlled quasi-experiment with 42 Bachelor students in the setting of a university learning environment. The method of experiments allows researchers to achieve high internal validity by carefully controlling the conditions under which an experiment is carried out [47]. The experiment was conducted

in two rounds, in which the students were divided into two groups and given different types of learning materials: conventional (as depicted in Figure 1) and context-based (Ref. Figure 6).

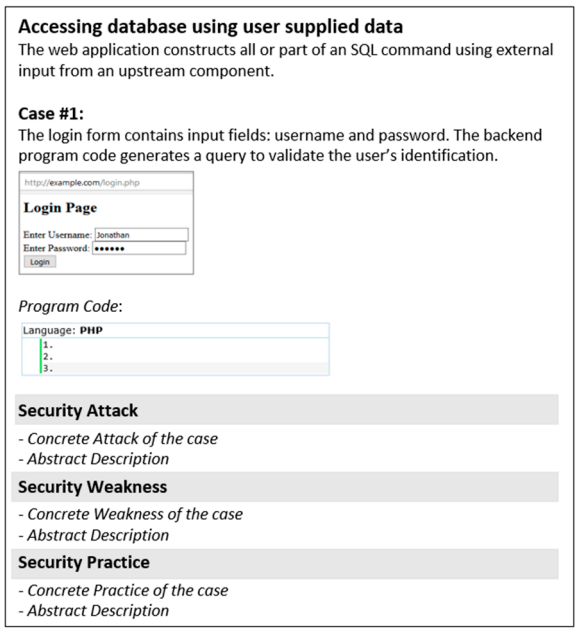


Figure 6. A simplified view of the learning material for SQL injection constructed using the proposed learning approach.

To evaluate the system's effectiveness, two instruments were employed: pre- and post-tests and survey questionnaires. Pre- and post-tests measure knowledge gain, while the latter instrument collects feedback responses. Table 1 presents the means analysis of the student's knowledge gained on the pre- and post-tests in each round of the experiment, including the mean scores and standard deviations. The results of the comparative means analysis show that there was a positive knowledge gain (i.e., post-test to pre-test score) for both groups in both rounds. However, the group using context-based learning materials had higher achievement levels than the group using conventional materials (Round 1: 9.54 vs. 2.75; Round 2: 10.91 vs. 4.50). To determine whether there was a significant difference between the initial knowledge of the two groups of students, we performed an independent sample *t*-test on the pre-test scores (Table 2). According to the statistical result, the significant (2-tailed) value in both two rounds is above 0.05 (0.137 and 0.534, respectively), which implies that there were no significant differences between the two groups in terms of the pre-test scores (i.e., the initial knowledge), and the significance of the knowledge gain can be concluded.

Table 1. Comparative means analysis of students' performance on the pre- and post-tests.

Round		Group A			Group B		
		N	Mean	SD	N	Mean	SD
1	Pre-test	20	26.75	5.20	22	24.32	5.19
	Post-test	20	29.50	6.90	22	33.86	4.86
	Knowledge gain		2.75			9.54	
2	Pre-test	20	21.75	8.78	22	20.00	9.26
	Post-test	20	26.25	6.90	22	30.91	8.54
	Knowledge gain		4.50			10.91	

Table 2. Independent sample *t*-test results for the post-test scores in the first round.

			Levine's Test		*t*-Test				
			F	Sig.	*t*	df	Sig. (2-Tailed)	Mean Difference	Std. Error Difference
Pre-Test	1st round	Equal variances assumed	0.238	0.628	1.516	40	0.137	2.432	1.604
		Equal variances not assumed			1.516	39.601	0.138	2.432	1.605
	2nd round	Equal variances assumed	0.012	0.913	0.627	40	0.534	1.750	2.791
		Equal variances not assumed			0.629	39.921	0.533	1.750	2.784

The learning satisfaction for the two learning materials is represented as a radar chart with six axes (Figure 7). The evaluation of learning satisfaction shows that the students showed higher overall learning satisfaction with the context-based knowledge approach than with conventional approaches. As depicted in the chart, the context-based learning material had overall higher learning satisfaction mean scores than the conventional materials in terms of the six satisfaction factors.

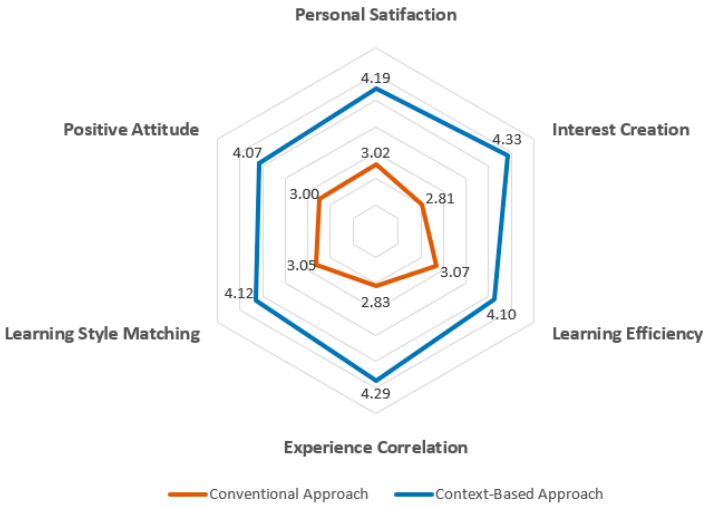


Figure 7. Learning satisfaction analysis.

4.2. DC 2: A Context-Based Ontology for Managing Contextualizing Security Knowledge

4.2.1. Objective and Solution

Taking the proposed learning approach into further design consideration, DC 2 focused on the artifact of the ontological knowledge model. Ontologies play the role of a binding factor that brings various knowledge items and processes together to provide a richer and more integrated view of the knowledge domain to the learners [48]. In our contextualized learning system, the ontology gives an explicit definition of the shared conceptualization of the software-security domain and assembling learning contents. It also supports effective knowledge acquisition and creation processes in the learning environment. Therefore, ontology is treated as a key component of the knowledge management perspective. In this regard, having a distinct design cycle to validate this component was necessary. The objectives of DC 3 were three-fold, (1) to design and construct an ontological knowledge base to manage contextualized knowledge, (2) to validate the feasibility of ontology, and (3) to visualize the knowledge representation as a pre-study for DC 3.

4.2.2. Design and Development

Our ontology is designed to facilitate linkages between software cases and contextual security knowledge by leveraging application contexts. From the security domain model, we use common vocabulary to denote meaningful concepts relevant to security. Depending on what is needed for the project, these terms can then be extracted and implemented into our ontology. The design of the ontology was divided into four perspectives, by the

strategies of said learning strategies: an application context perspective, software technology perspective, abstract security knowledge perspective, and contextualized security knowledge perspective. The overall ontology of the four perspectives is illustrated in Figure 8, and the details of each perspective are described below.

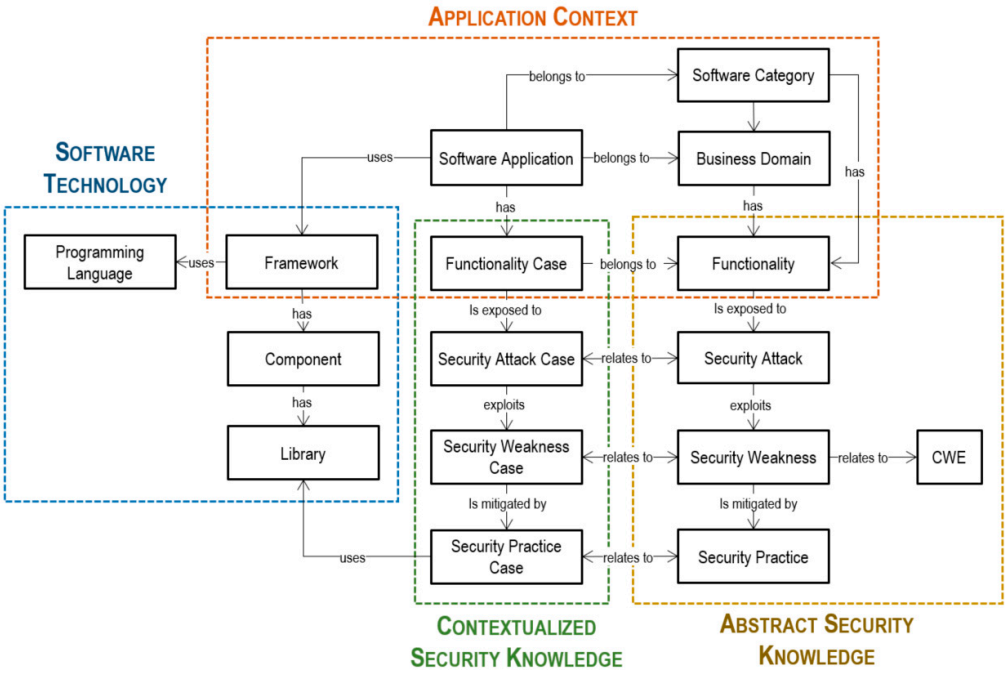


Figure 8. The context-based security ontology.

- Application context perspective.

Security knowledge contextualization is supported by the creation of contextual cases in different application contexts. The application context is a collection of characteristics, which describe the properties of a software application. In the ontology, we model characteristics that are highly relevant for retrieval of security knowledge within a software application, concerning four perspectives: Software Category, Business Domain, Software Technology, and Functionality.

- Software technology perspective.

The software technology perspective describes techniques and tools that support building and running a software application, covering Programming Language, Framework, Component, and Library. This perspective is constructed as a technology repository, which includes elements in defining the application context (i.e., the used programming language and the technology framework) and, meanwhile, provides reusable resources (i.e., technology libraries) in describing contextualized security knowledge.

- Abstract security knowledge perspective.

The abstract security knowledge perspective describes the security knowledge that is an object of teaching through a set of concepts (topics to be taught). We identified three security concepts that are most widely used throughout the security domain and need to be concentrated on learning about security attacks, security weaknesses, and security practices. From a security conceptualization point of view, we only want to indicate which principles or abstract ideas are needed, not their practical implementation. Therefore, we describe security knowledge in this model at a level of abstraction. The instances of these classes specify only the fundamental characteristics of the security concepts, not specific software application aspects. The main advantage of this design is to share a common understanding of conceptual security knowledge among different security contexts.

- Contextualized security knowledge perspective.

The abstract security knowledge needs to be put in context so that it can adapt itself to different situations of software development. In this regard, security contextualization knowledge describes security knowledge in the context of a specific case and brings together abstract knowledge. The included security concepts are aligned with those defined in the security domain model, which are security attacks, security weaknesses, and security practices. The term contextualization is used here to describe the process of drawing specific connections between security domain knowledge being taught and an application context in which the conceptual knowledge can be relevantly applied or illustrated.

4.2.3. Demonstration and Evaluation

The ontology was constructed and demonstrated by Protégé Editor [49] with Ontology Web Language (OWL) and then validated through a three-phase evaluation process [50]. First, the ontology structure, including concept definitions and relations, was reviewed and analyzed by an internal security professional within NTNU who provided the competencies using a computer/cybersecurity and ontology building method and analysis. The ontology structure, including concept definitions and relations, were reviewed and analyzed. In phase 2, the ontology was evaluated by answering competency questions against its initial requirements. One exemplary question is: *What are the available software scenarios given in the functionality "Generating output in web pages using user-supplied data", PHP language, and MySQL database?* When searching the ontology, we use the SPARQL protocol [51] to extract information from the RDF graph. After the domain expert's review and a competency-question examination, we took a further application-based evaluation approach [52,53] by plugging the ontology into an application for validating its applicability.

4.3. DC 3: An Ontology-Based Contextualized Learning System for Software Security

4.3.1. Objective and Solution

The third design cycle aimed to construct a contextualized learning system for software security. This proposed framework acted as a proof-of-concept for security educators, demonstrating an ontology-driven web application that could enable context-based learning. The artifacts designed in DC 1 and DC 2 were integrated to create the appropriate system: the formerly provided representation of security knowledge and an embedded learning process, while the latter acted as the kernel knowledge base.

4.3.2. Design and Development

This proof-of-concept learning system featured a two-tier system architecture, depicted in Figure 9. The front end was a web-based user interface built with HTML, JavaScript, and JQuery. This is where users interacted with their learning content. On the backend, an Apache Tomcat server supported Java technologies like Servlets, JSP, and AJAX to manage application logic. For the implementation of our security knowledge base, we employed ontology technologies. An administrator used Protégé Editor to create and manage the ontology, and an OWL file was hosted on a web server. The use of these tools enabled us to successfully develop our proposed security knowledge base. To extract data from the ontology repository, Apache Jena API (https://jena.apache.org/, accessed on 2 November 2022) was chosen to be the core technology. Jena is an open-source Semantic Web framework for Java. It helps developers develop code that handles Semantic Web building blocks such as RDF and OWL in line with published W3C recommendations (https://www.w3.org/2001/sw/, accessed on 2 November 2022).

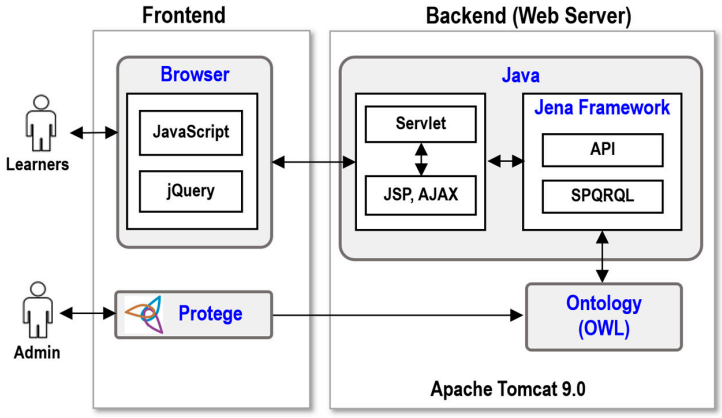


Figure 9. System architecture diagram.

The prototype system's user interface is depicted in Figure 10, showcasing HTML output in a web application context. The educational process commences with concrete information that learners are familiar with, eventually morphing into comprehension of abstract concepts. This scenario serves as the launch point for teaching security concepts to users on a need-to-know basis and displaying the exemplified knowledge. Depending on the sought-after knowledge, learners choose pertinent criteria from the application-context menu to tailor the learning environment.

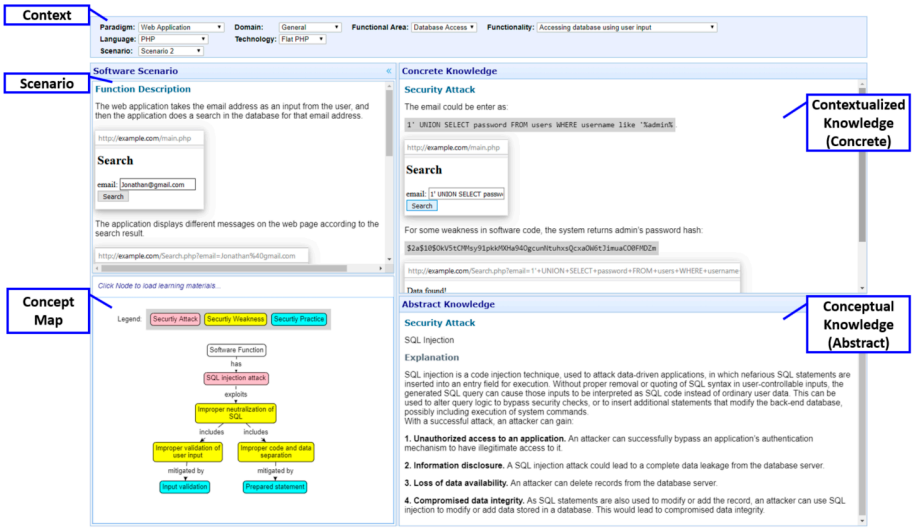


Figure 10. The proof-of-concept learning system.

To help learners access contextualized knowledge in a timely and effective manner, they must understand the relationship between security concepts. On the one hand, they should be able to grasp which causes and effects are pertinent to the material they are studying. This clarity of association allows learners to comprehend their learning content.

On the other hand, students must be able to synthesize the semantic value of knowledge structures into their cognitive schemata to facilitate efficient learning. For this purpose, we outline the learning contents in a graphical Concept Map, which shows in the left-corner part of the screen. The concept map utilizes a concrete-to-abstract presentation structure, making relevant knowledge easily accessible. When clicking on a node, detailed information is made available on the right side of the screen. The top portion displays contextualized knowledge, while underneath it offers an abstract explanation for further clarity. Concrete representations can be used to develop a stronger understanding of security domain knowledge. These could include realistic simulations of exploits and attacks, identifying mistakes in source code, and learning about secure coding practices

for remediation. Learners progress from the concrete to the abstract as they move through their educative experience, with dynamic application scenarios integrated into the security domain knowledge.

4.3.3. Demonstration and Evaluation

- Preliminary evaluation.

An experiment in a university learning environment was conducted to evaluate the effectiveness of the proposed learning system. The participants were 36 Bachelor's students from two main study programs, IT operations in information security and Programming. The participants were randomly assigned to either control or experimental groups. The students in the experimental groups were treated with the proposed learning system while the control group adopted conventional learning materials. This experiment employed pre-test/post-test and questionnaires to measure students' security knowledge gain and their learning satisfaction, respectively.

During the experiment, two groups of students (18 students each) took 15 min to complete the pre-test sheet before studying the learning materials using the treatments assigned to them. The learning session lasted for one hour. After completing the learning session, which lasted for one hour, all students participated in a post-test exam. In addition to the exam, students who used the learning system also completed questionnaires. Table 3 reveals the mean analysis of students' performance on the pre- and post-test, including the mean scores and knowledge gain. Students' learning satisfaction with using the learning system is presented in Table 4. The result of the experimental evaluation showed its usefulness and feasibility because it shed some light on the potential benefits of context-based learning. First, with the support of contextualized learning, the experimental group students yielded significantly better performance than those in the control group in terms of security knowledge gain. Second, according to the results of the questionnaire survey, the students expressed higher learning satisfaction with the learning system using contextualized security knowledge than conventional learning materials. Moreover, most students were very interested in the proposed learning system, and all agreed that this approach could ease the information load effectively.

Table 3. The evaluation of students' learning satisfaction with the learning system.

		Mean	Std. Deviation
Control Group	Pre-Test	30.00	11.757
	Post-Test	40.28	9.922
	Knowledge Gain	10.28	
Experimental Group	Pre-Test	30.83	11.789
	Post-Test	46.94	7.503
	Knowledge Gain	16.11	

Table 4. The evaluation of students' learning satisfaction with the learning system.

Category	Question	Mean
System Operation	I agree that the applied learning technique in the system is novel and it can assist my learning.	4.11
	I am very clear about the learning procedure embedded in the system.	4.00
	The system organizes security knowledge in a structured and collected manner.	4.21
	The knowledge content provided by the system is easy to understand.	4.00
	I think that the system is useful for learning security knowledge.	4.05
	Average	4.07

Table 4. *Cont.*

Category	Question	Mean
Learning Attitude	The system helps me deepen the memorized impression of the learning subject.	4.11
	The system helps me relate security knowledge to what I knew or experienced before.	4.16
	The system reduces the difficulty of learning secure programming.	4.11
	I find that at times studying the learning materials gives me a feeling of personal satisfaction.	4.05
	The system helps me foster a positive attitude toward learning security knowledge.	4.00
	Average	4.09

- Evaluation in open-source software development environments.

As the concept of software development has changed over the past few years, it is important to look into research topics within the field in which the software security learning process is embedded and implemented. Examining specific fields critically can help us in making informed decisions regarding software security. In this regard, the context of open-source software (OSS) development was chosen. OSS has had a growing impact on society and today's ICT systems: approximately 80% of companies run their operations on OSS [54], and 96% of applications utilize OSS as software components [55]. The Linux Foundation (the Linux Foundation (LF) is a non-profit technology consortium founded in 2000 as a merger between Open Source Development Labs and the Free Standards Group to standardize Linux) reported that the Linux kernel has committed over 25 million lines of code from over 33,000 open source contributors [3]. However, over 80% of OSS project maintainers and users believe developers should own security, but they are not well-equipped, according to the State of Open Source Security Report—2019 [56].

The objectives of the fourth design cycle were to test and validate the beta version of the learning system with software developers in OSS development projects. We aimed to answer the research question: *To what extent does the proposed security learning system affect the learning outcome in OSS development environments?*

The ontology was prepared with real-world scenarios from a web application, specifically an e-Store system with development in both PHP and Java. The web application was created by the author to facilitate demonstration and learning material creation. The learning system was then deployed to the internet and tested by open-source developers. The evaluation phase included an online questionnaire to capture individual user perceptions of the learning system. Participants provided both quantitative and qualitative responses regarding system features, embedded learning approach, weaknesses, and strengths of the system. Twenty-one voluntary participants from GitHub responded to our research invitation letters and completed this survey questionnaire. IBM SPSS software was used to assess the reliability of the data utilizing Cronbach's alpha for internal consistency of the evaluation items, confirming that the scale in question is unidimensional, depicted in Figure 11. The derived alpha value was 0.834, which was above the acceptable threshold (0.70) suggested by Numally [57]. Thus, the survey items on the instrument are deemed highly reliable and appropriate for such research.

Case Processing Summary

		N	%
Cases	Valid	21	100.0
	Excluded[a]	0	.0
	Total	21	100.0

a. Listwise deletion based on all variables in the procedure.

Reliability Statistics

Cronbach's Alpha	N of Items
.834	5

Figure 11. SPSS reliability test of evaluation items within the category of "Learning approach".

As evidenced by Figure 12, the learning approach proved successful: 91% of respondents agreed that it was an efficient way to learn software security, while 85% felt the conditions were right for effective learning. Additionally, 80% reported being drawn to this approach and enjoyed learning software security through it. These results are encouraging and demonstrate the effectiveness of this learning approach. Further analysis of the feedback collected from respondents showed they highlighted the practicality of system features and positively recommended its use with software scenarios featuring interactive and contextualized security knowledge presentations. Furthermore, the approach was deemed successful in keeping learners engaged, with positive satisfaction ratings given for learning sessions. The results presented in this paper demonstrate that a context-based approach is suitable for supporting developers' training and education on software projects.

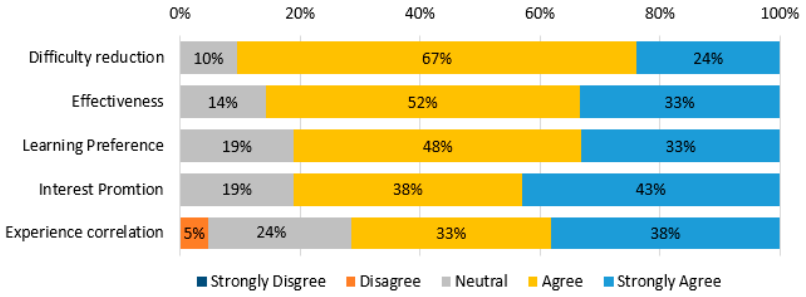


Figure 12. Responses to the questions about the proposed learning approach.

4.4. DC 4: Revision of the Learning System and an Evaluation in a Software Company

4.4.1. Objective and Solution

Our past two empirical studies provided promising indications of the effectiveness of a context-based learning approach in both educational and OSS development settings, with evidence that the proposed learning system could motivate developers to learn more about software security. Continuing this research, we conducted evaluations in an industrial setting. Working with a Norway-based software development company, we aimed to validate and examine our learning system, along with the contextualized learning it incorporated.

4.4.2. Design and Development

For demonstration purposes, we refined and released a new learning system titled "CLEAR," an acronym for "Contextualized Learning System". The revised user interface of CLEAR consists of three main parts: (a) Main menu, (b) Filter, and (c) Knowledge content. The main menu provides different perspectives for learners to access context-based security knowledge, with perspectives that are both contextual case and technology-based. Figure 13 depicts the system from the perspective of the contextual case.

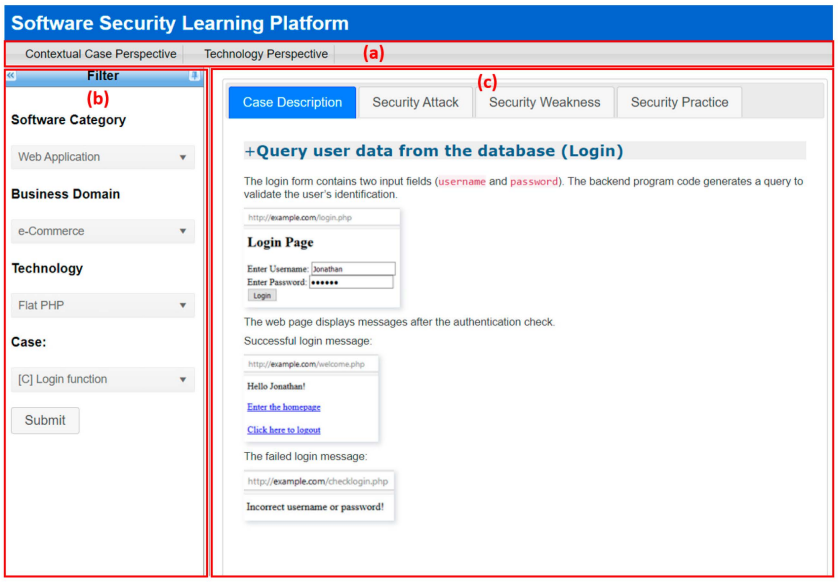


Figure 13. CLEAR user interface (Contextual case perspective): (a) Main menu, (b) Filter, and (c) Knowledge content.

A. Contextual case perspective

Contextual cases remain the primary source of security knowledge gained using our platform. Selection criteria are used to guide learners in categorizing learning sessions and filtering contextual cases for desired knowledge. To do this, a multi-level tree view menu displays functionality cases, including authentication functions, searches, file operations, and more. Used as an anchor event for security-learning retrieval, the contextual case is visible throughout each learning session to help the learner make connections between their experience and the new information they gain.

Our learning content includes four tabs: Case Description, Security Attack, Security Weakness, and Security Practice. These tabs feature practical demonstrations of relevant application functions that take learners' prior experience into account. The three security-related knowledge tabs follow an organized order with one another; the abstract explanation of each learning topic can be found beneath the contextualized content. Moreover, our learning material is comprehensive and extensive; it consists of perceptually rich details such as showcasing how security attacks work with various exploits, pinpointing common mistakes in code, and giving secure coding practices to fix them.

Figure 14 illustrates the material shown in the three tabs. Through a focused, straightforward approach to presenting security-related knowledge, we can help guide learners as they pass through various use cases by consistently asking three key questions: What are the possible attacks? Why does it experience these attacks? How can we prevent them? Answering these what–why–how questions encourage learners to form connections between different security concepts and construct accurate mental models. Such mental models, essential for effective learning, help develop focus and organization of complex concepts [58]. After gaining general knowledge from the tabs, users can access more detailed conceptual knowledge by clicking on titles. An additional mouseover text ("Click to expand details") has been added above each title to easily indicate which pieces of material offer further insight (Figure 15). This combination of practical and conceptual information helps make sense of contextualized security knowledge within its relevant domain.

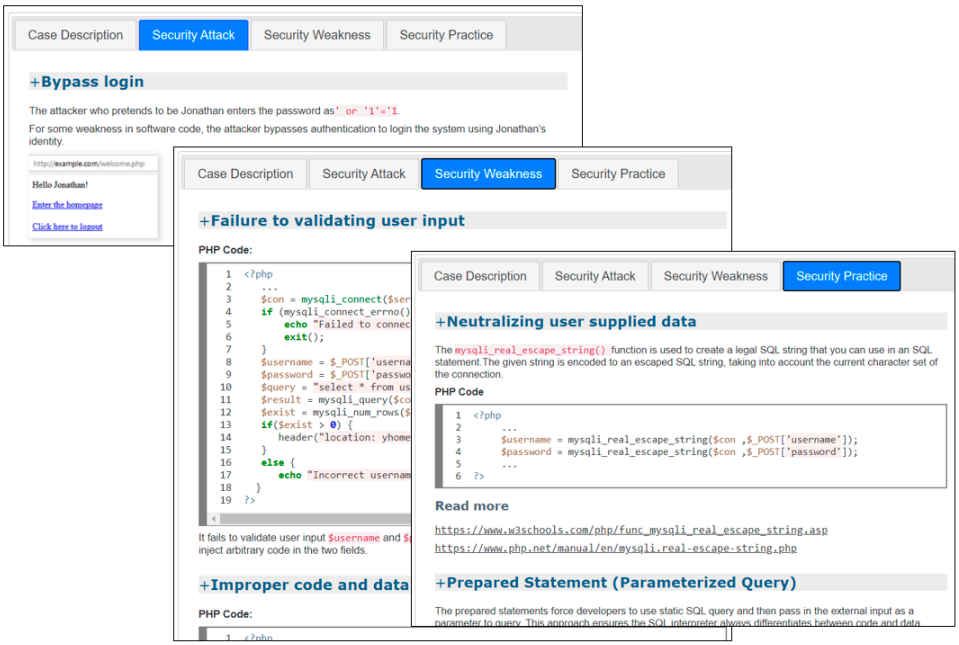


Figure 14. Security knowledge presentation for a contextual case.

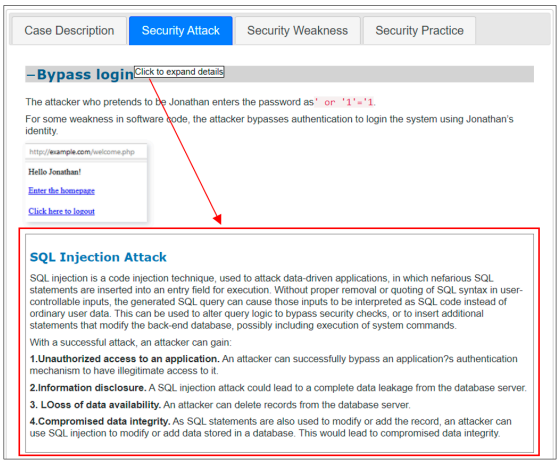


Figure 15. Security knowledge presentation for a contextual case.

B. Technology perspective

The technology perspective provides learners with advice about software security practices tailored to the systems, tools, and technologies they are using or considering. Figure 16 shows a detailed view of the system from this point of view. We use the OWASP Secure Coding Practices Quick Reference Guide [59] as our source material, which outlines 14 key areas such as input validation, output encoding, session management, and more. To make these concepts easier to understand, we have broken them down into two groups: components and libraries. With an emphasis on coding best practices and emerging technologies, this approach teaches an invaluable skill set for any learner interested in software security.

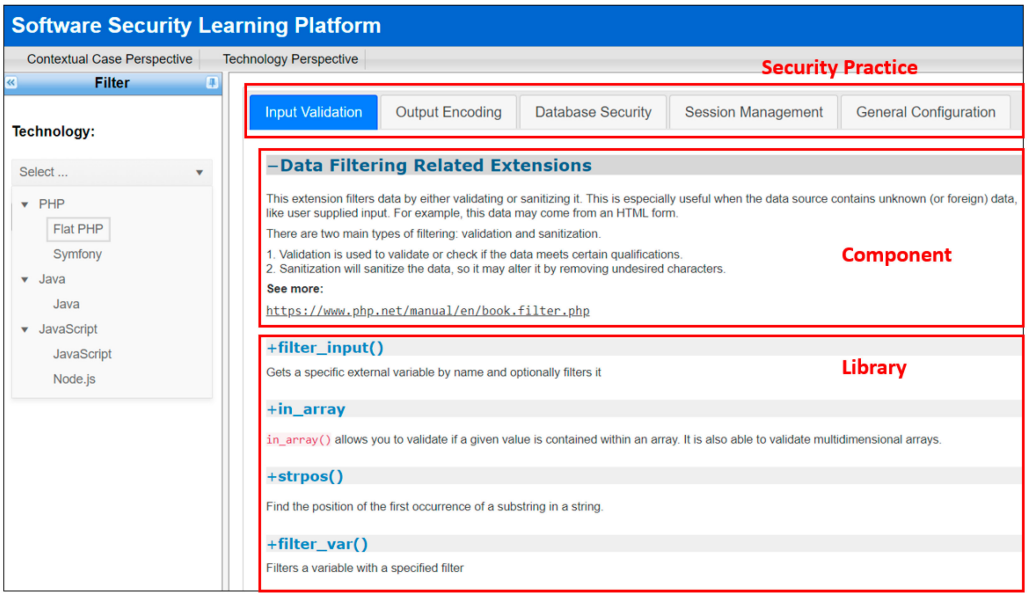


Figure 16. System user interface (Technology perspective).

The *component* is the unit of functionalities described at an abstraction level, while the *library* represents a collection of toolkits, functions, reusable code, and APIs used to implement security practices. For example, in the technology of Flat PHP, developers can use libraries filter_input() or filter_var() in the component of data filtering extensions to validate user input. Within libraries, several concrete examples with sample code are provided, which can be expanded and collapsed dynamically by learners (Figure 17). Researchers indicate that programmers learning how to use a programming language or API often rely on code examples to support their learning activities [60,61]. Code examples can come to their help and lift the burden of solving problems on their own. Figure 18 presents the screenshots for security practices of output encoding, data security, and session management.

Figure 17. Details of technology libraries.

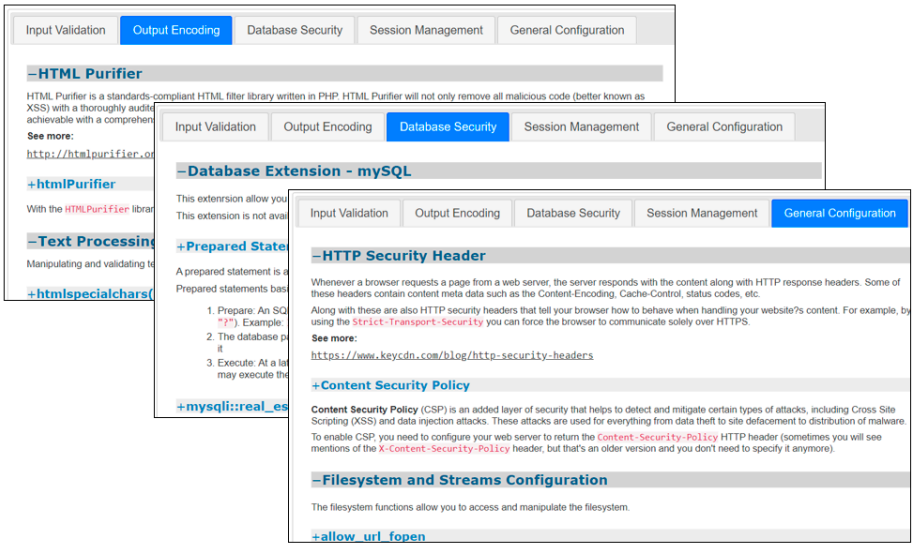


Figure 18. Security knowledge presentation from the technology perspective.

C. Demonstration and evaluation

For system demonstration, we first prepared eight cases under five critical web application functionalities. Each case corresponds to a specific security attack. The relationship between the functionalities, cases, and the corresponding security attacks is presented in Table 5.

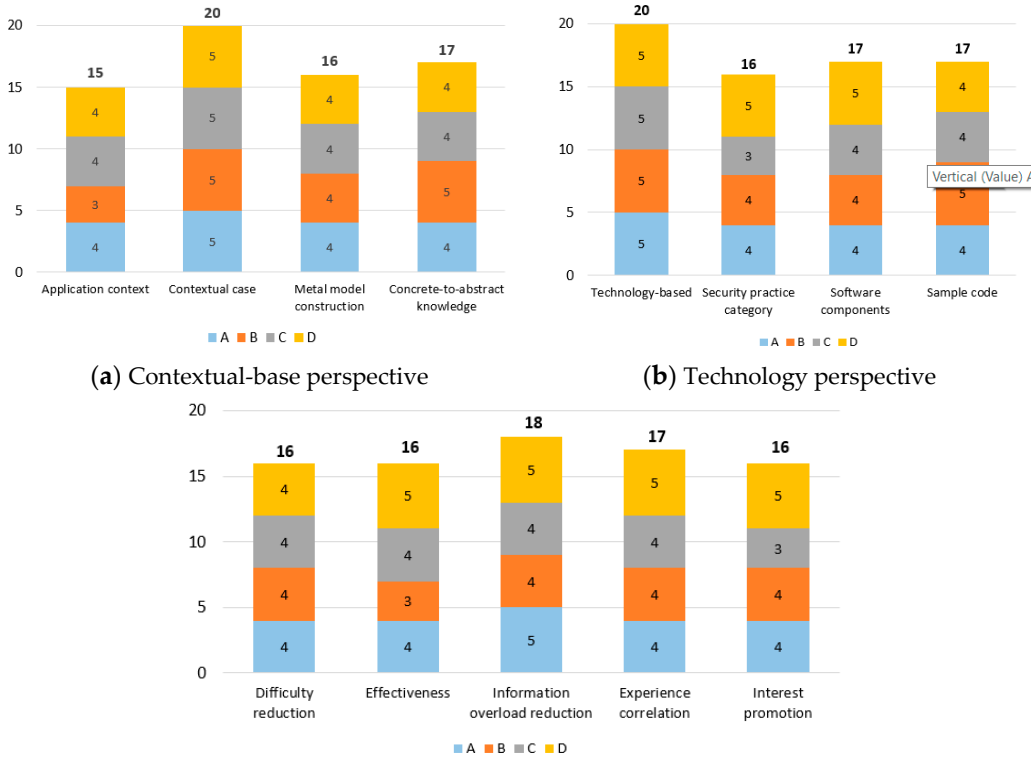
Table 5. The configuration of functions/cases and corresponding security attacks.

Functionality	Case	Security Attack
Generate HTML using users supplied data	User Registration	Cross-site scripting
	Bulletin board	Cross-site scripting
Functions with users' credentials	Change password	Cross-site request forgery
	Update Profile	Cross-site request forgery
Access database using user-supplied data	User login	SQL injection
	Search email	SQL injection
Include external files in HTML	Switch displayed language	Remote file inclusion
File operation	List documents	Directory traversal

To understand how a proposed learning system would play out in an industrial setting, we conducted a case study. Case studies have the advantage of being based on real-world scenarios, rather than purely theoretical ones. In this way, they can help bridge the gap between proof-of-concept and actual use or value assessment [62]. The case study was conducted at Escio, a software development company located in Norway. We recruited four developers from Escio who were actively pursuing programming projects; all four were male and had programming experience that ranged from one to ten years using primarily PHP and JavaScript languages. Our objectives for the case study were twofold: (1) to evaluate and test an alpha version of the learning system with software developers and (2) explore the effectiveness of the proposed security learning system as well as any lessons learned.

This case study used a Mixed-Methods Research (MMR) approach in the evaluation phase, which combines quantitative and qualitative approaches. We designed an online questionnaire with a 5-point Likert scale to collect data on system user experience from developers. Due to the small participant pool, these results may not be generalizable across the population; thus, semi-structured interviews were arranged for further qualitative and interpretive analysis. This interview format was more relaxed because the interviewer could follow up on answers and topics of interest. All participants found this approach useful for providing their insights.

The evaluation period for this trial system and the learning sessions was kept short due to the schedules of employees and the company constraints. After completion, employee feedback was collected via questionnaires and interviews. Figure 19 presents their responses to various evaluation items. Particularly, they all had positive reviews on the contextual case element (Figure 19a). It garnered a score of 20 out of all evaluations—showing its practicality to users. In interviews, many participants highlighted how having a clear software functionality that comes with corresponding security knowledge is key to learning about software security. Here are some reflections from those interviewed:



(**a**) Contextual-base perspective

(**b**) Technology perspective

(**c**) Overall system satisfaction

Figure 19. Quantitative analysis in DC 4.

> *"This is a good design where people can learn things from real-world cases. Secure programming is something that you will have to understand and experience a lot. If it comes to learning it from real-time examples, then it's very easy as you can understand things from your perspective."*
>
> *"We [developers] learned some surface features of security principles and concepts of how we were taught in the school. In works, however, we need to have enough experience to understand its application. It is very important to provide such learning experiences that can help us solve security problems in different contexts."*
>
> *"I appreciate how it connects software behavior with relevant security information—it's excellent for learning."*

Participants noted the effectiveness of introducing security concepts through concrete-to-abstract strategies when conveying information. They commented:

> *"When it comes to complex coding solutions, they are more easily comprehended when the solution is presented with accompanying example code. It's an easier way for readers to understand how something works in comparison to reading long paragraphs of explanation."*
>
> *"I found it easier to understand the information when it was presented in a step-by-step, concrete manner."*

"I liked how the presenter used analogies and comparisons to help us understand the importance of security measures."

Figure 19b in the survey results indicates that all four items had a mean value of 4 and that technology-driven knowledge presentation earned a maximum score of 20. Several user comments confirmed this strong showing:

"The design of putting [programming] language and the [secure] coding practices together servers one-stop learning for me."

"I prefer to understand [software] security by studying the given practices of the programming language. Especially nowadays, we developers, commonly use libraries or functions for daily tasks. Knowing the security knowledge from the technology side is straightforward and very efficient."

The demonstration of security practices with software components and libraries bundled with sample code during the course was met with high praise and appreciation from the participants. Comments such as:

"The short snippets of source code and organization of the content made it easily digestible."

Lastly, the results of Figure 19c show that participants were satisfied with the features of the system. They felt that it alleviated information overload when they were learning about software security and also allowed them to connect their prior knowledge and experiences to concepts. All participants strongly agreed with the statement "The system helps me relate security knowledge to what I knew or experienced before."

Some comments about system satisfaction are listed below:

"The way of interaction is relatively simple: the designed frames lead to sufficient visual space. Users do not need to swap between different pages."

"Easy to use. Good information. Good readability and layout."

5. Discussion

The merits and benefits of conducting design scientific research can be numerous for the researcher, the community, the practice, and ever-developing science. This research contributes to the field of software development and security education. In particular, it sheds light on context-based support in teaching and learning software security from the aspects of learning approaches and knowledge modeling. This section concisely describes the contributions in terms of the initial goals and the research questions.

RQ 1: How can context-based approaches be applied in software security to motivate learners and improve learning outcomes?

This research question addresses the DC 1 of DSR activities, seeking to develop a novel method for software security instructional design and teaching. For this, we propose a context-based approach that features three strategies: (1) starting from a meaningful scenario, (2) stimulating mental models for learning, and (3) moving from abstract to concrete knowledge. This research project investigated the effectiveness of a context-based learning approach in the university setting through a quasi-experiment with 42 Bachelor students. The findings showed that this approach had many positive effects, such as improved academic performance and student attitude toward software security. In addition, the study revealed that context-based instruction was more useful for practical problem-solving. Overall, context-based instruction can be an effective solution when teaching courses related to software security or computer science.

RQ 2: What approach can be taken to develop an ontology that effectively manages contextualized knowledge related to software security?

This research examines the efficacy of using an ontology to manage contextualized knowledge in software security. Our research developed a two-model approach that separates concrete and abstract knowledge; this hones maintenance and retrieval while simultaneously enabling semantic interoperability. Additionally, we built the ontology

with OWL and Protégé, allowing its use by automated tools which can provide advanced services for security requirements and design recommendations. This ontology stands out from other existing works in its context-oriented nature. It captures security information based on a wide range of software features and technologies, while also providing abstract explanations for concrete data. Additionally, ontology models security knowledge within a contextual framework. With this ontology-based system, we hope to address existing issues in the software security domain to provide better learning opportunities.

RQ 3: What strategies can be employed to build a learning system for software security that incorporates contextualized content, and what impact does this system have on the overall learning outcomes?

This research identified the need for a contextualized learning system that could enhance software security learning. The development of this essential learning artifact forms an integral part of this work and is key to providing more effective learning tools in this area. This learning system integrated the effective-proof context-based learning approach and the ontological knowledge base to facilitate the contextual retrieval of security knowledge and contextualized learning. In such an environment, learners discover meaningful relationships between the abstract explanation and the practical demonstration in the context of real software applications they are already familiar with; security concepts are internalized through the process of discovering, reinforcing, and relating.

The learning system was evaluated through a comprehensive process, including quantifiable and qualitative testing with three separate groups: Bachelor students, OSS developers, and industrial developers. The evaluations yielded favorable results, suggesting that the proposed learning system has the potential to be an influential teaching tool that can encourage potential users (e.g., existing and future developers) to become educated on software security. Our research study showed that the proposed contextualized learning system, CLEAR, has the potential to be an invaluable tool for software developers. We found that participants appreciated the practicality and context-based approach of CLEAR, which kept them interested and engaged. Furthermore, we observed all participants achieved positive knowledge gain after using it, allowing them to effectively resolve security issues.

6. Threats to Validity

Though this research has achieved promising results, it is important to consider its limitations. First, our context-based approach to security learning is limited in terms of completeness and representativeness, as the strategies and methods used were identified based on a literature review. We tried to provide an accurate representation of context-based learning in the field of software security; however, the understanding will always be subjective. Since this research is based on the qualitative analysis performed by the author, the results may be biased due to the individual's background, culture, and personal experiences. Given the qualitative nature of this research, it is essential to perform multiple reviews of the learning approach. Second, the experiment was relatively short, lasting only around 40 min per session with immediate assessment of learning outcomes. Unfortunately, any long-term effects or knowledge retention remain unclear; extended time frames and further assessments may be needed to assess the efficacy of learning performance over a longer period.

The second limitation stems from the evaluation of the contextualized learning system. The artifact developed as part of this research has been proven to be very successful, as acknowledged by a three-phase evaluation process—an experiment in the school setting (with bachelor students), a survey with OSS developers, and a case study in an industrial context—the promising results may still be somewhat biased. First, this experimental evaluation took place at a university with student volunteers enrolled in the Software Security course. This particular sample is not necessarily representative of the population as a whole, and the limited sample size (36) also hindered its generalizability. Second, this study addressed the evaluation of a system in several real-world software development

contexts, with the findings based on self-reported experiences and perceptions regarding the proposed learning system. To ensure the accuracy of these results, it is necessary to consider the issue of retrospective reporting. The ability to assess true interaction between individuals and this learning system or determine actual engagement time cannot be determined with self-reporting alone. When considering the data collected through the survey, it should be noted that the sample size was not sufficient; with so many developers active today, a larger pool of participants would offer more valuable insights. To gain a better understanding of software-project dynamics, replication studies are recommended with larger sample populations.

7. Conclusions

This paper seeks to explore education and learning surrounding software security. Our goal is to provide an overview of existing approaches in the domain and to identify promising directions for future research related to context-based security learning and knowledge management strategies. Specifically, this research utilizes the Design Science Research Methodology (DSRM) and employs a four-cycle approach to ensure the integration of vital components in the creation of a context-based learning system. The aim is to restructure security knowledge and establish an effective context-based learning process that developers and other learners can utilize to enhance their knowledge of software security. This research work includes a comprehensive evaluation of the system's effectiveness in helping developers achieve expected learning outcomes and learning satisfaction. This evaluation relied on the perspectives of potential users, including students and software developers. Overall, this research presents concrete evidence that changing structures in security knowledge transfer ultimately make it easier for developers to access and apply concepts in various settings—an invaluable advantage for readily understanding mobile app security information.

Context is critical for bridging the gap between what developers know and what they need to understand about security. This finding tells us that when security knowledge is presented in real-world contexts, it resonates with developers' existing programming experience, thus sparking their enthusiasm for learning. By altering the sequence of security knowledge transfer, it is possible to motivate learners more effectively. Contextualized learning approaches may be used to make security training engaging and purposeful; at the same time, having a well-designed learning system can help support the further promotion of this concept.

This contextualized learning technique has produced some promising results. Students appreciated the "learning journey" aspect of the approach, and many software developers praised the research for its ability to generate motivation in learning security knowledge. This encouraging feedback bodes well for context-based learning methods in the field of security education and training. Software professionals and educators are welcome to test these complementary approaches to achieve a heightened understanding among public audiences. We look forward to continuing our work on this system as we strive to advance accessible security knowledge in both academic and industry settings.

7.1. Future Research Opportunities

There are numerous possible research opportunities in both these areas which could prove to be very interesting and may yield great insights if studied. Moving forward, we should consider developing the learning system further by exploring each of the outlined potential research opportunities.

7.1.1. The Context-Based Learning Approach

Context-based methods present a new and innovative approach to software security education, which has not undergone in-depth research until now. These techniques provide the potential to bring major advancements to the surrounding field; thus, security education could be greatly beneficial from this cutting-edge method of learning. The proposed

research in this area looks extremely promising. This research outlines a context-based approach that could be adapted to other educational spheres, such as information security and computer security. Furthermore, studies on learning processes and factors potentially influencing learning processes are still needed to develop fine-grained models of context-based security learning. Future research could also examine the approach in combination with the proposed teaching strategies to pilot context-based learning at various points of the security instruction process, for example, the design of lectures and instructional materials. Moreover, studying the long-term impact and effectiveness of context-based learning approaches in the domain of software security would provide valuable insights. Longitudinal studies can track knowledge retention, practical application, and overall improvement in software security practices among learners who have undergone context-based training. Further investigation can be conducted to explore the effectiveness of incorporating more real-world security scenarios and practical examples into the learning process. This would entail the creation of contextualized learning materials that leverage artifacts generated during the software development process, such as code review results, vulnerability scan results, penetration test findings, and other relevant data.

7.1.2. The Contextualized Learning System

This research resulted in the development of a contextualized learning system intended to boost a context-based understanding of software security. Furthermore, research can focus on developing intelligent adaptive learning systems that personalize the learning experience based on individual learners' needs, prior knowledge, and learning styles. By tailoring the content and delivery of software security training to each learner's context, such systems can enhance knowledge retention and skill acquisition. Once complete, the enhanced system will offer a comprehensive set of content for learners at various levels. Furthermore, this research did not incorporate socio-technical methods to examine the impact this artifact may have on organizational culture and structural formation [63–65], which are usually used to evaluate the outcomes of new technologies at existing learning programs. Researchers have an opportunity to close the knowledge gap by using the learning system in OSS development environments. This system can provide security knowledge while allowing researchers to observe changes related to security culture and software quality. Further research with this model could also improve its usability, user interfaces, and maintainability of security knowledge.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: All parties consented to having their names and statements published in this paper.

Data Availability Statement: Please email the author for access to generated data.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Stack Overflow. Developer Survey. 2021. Available online: https://insights.stackoverflow.com/survey/2021 (accessed on 2 November 2022).
2. Veracode; DevOps.com. DevSecOps Global Skills Survey. 2017. Available online: https://info.veracode.com/analyst-report-devsecops-global-skill-survey.html (accessed on 2 November 2012).
3. Larabel, M. The Linux Kernel Has Grown by 225k Lines of Code So Far This Year from 3.3k Developers. 2018. Available online: https://www.phoronix.com/scan.php?page=news_item&px=Linux-September-2018-Stats (accessed on 4 November 2020).
4. Agrawal, A.; Rahman, A.; Krishna, R.; Sobran, A.; Menzies, T. We Don't Need Another Hero? The Impact of "Heroes" on Software Development. *arXiv* **2017**, arXiv:1710.09055.
5. Young, A. Too Much Information: Ineffective Intelligence Collection. 2013. Available online: https://hir.harvard.edu/too-much-information/ (accessed on 13 October 2012).
6. Hoq, K.M.G. Information overload: Causes, consequences and remedies-A study. *Philos. Prog.* **2014**, *55–56*, 49–68. [CrossRef]
7. Murayama, K.; Blake, A.B.; Kerr, T.; Castel, A.D. When enough is not enough: Information overload and metacognitive decisions to stop studying information. *J. Exp. Psychol. Learn. Mem. Cogn.* **2016**, *42*, 914. [CrossRef]

8. Ryoo, J.; Techatassanasoontorn, A.; Lee, D. Security education using second life. *IEEE Secur. Priv.* **2009**, *7*, 71–74. [CrossRef]
9. Apvrille, A.; Pourzandi, M. Secure software development by example. *IEEE Secur. Priv.* **2005**, *3*, 10–17. [CrossRef]
10. Fenstermacher, G.D.; Richardson, V. On making determinations of quality in teaching. *Teach. Coll. Rec.* **2005**, *107*, 186–213. [CrossRef]
11. Land, S.; Jonassen, D. *Theoretical Foundations of Learning Environments*; Routledge: London, UK, 2012.
12. Cooper, S.; Cunningham, S. Teaching computer science in context. *ACM Inroads* **2010**, *1*, 5–8. [CrossRef]
13. Ko, A.J.; Myers, B.A. Debugging reinvented: Asking and answering why and why not questions about program behavior. In Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany, 10–18 May 2008; ACM: New York, NY, USA, 2008; pp. 301–310.
14. Gruber, T.R. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.* **1995**, *43*, 907–928. [CrossRef]
15. McGraw, G. *Software Security: Building Security In*; Addison-Wesley Professional: Boston, MA, USA, 2006.
16. Dey, A.K. Understanding and using context. *Pers. Ubiquitous Comput.* **2001**, *5*, 4–7. [CrossRef]
17. Brézillon, P. Modeling and using context: Past, present and future. In *Rapport de Recherche Interne LIP6*; University of Paris: Paris, France, 2002.
18. Brézillon, P.; Araujo, R. Reinforcing shared context to improve collaboration. *Rev. Sci. Technol. Inf.-Série RIA Rev. Intell. Artif.* **2005**, *19*, 537–556. [CrossRef]
19. Ackerman, M.S. Definitional and contextual issues in organizational and group memories. *Inf. Technol. People* **1996**, *9*, 10–24. [CrossRef]
20. National Research Council. Evaluating and improving undergraduate teaching. In *Science, Technology, Engineering, and Mathematics*; National Academies Press: Washington, DC, USA, 2003.
21. Brézillon, P. Making context explicit in communicating objects. In *Communicating with Smart Objects: Developing Technology for Usable Pervasive Computing Systems*; Kogan Page: London, UK, 2003.
22. Brézillon, P. Context in problem solving: A survey. *Knowl. Eng. Rev.* **1999**, *14*, 47–80. [CrossRef]
23. Perin, D. Facilitating student learning through contextualization: A review of evidence. *Community Coll. Rev.* **2011**, *39*, 268–295. [CrossRef]
24. Predmore, S.R. Putting it into Context. *Tech. Connect. Educ. Careers* **2005**, *80*, 22–25.
25. National Research Council. *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*; National Academies Press: Washington, DC, USA, 2000.
26. Bennett, J.; Lubben, F.; Hogarth, S. Bringing science to life: A synthesis of the research evidence on the effects of context-based and STS approaches to science teaching. *Sci. Educ.* **2007**, *91*, 347–370. [CrossRef]
27. Gilbert, J.K. On the nature of "context" in chemical education. *Int. J. Sci. Educ.* **2006**, *28*, 957–976. [CrossRef]
28. Shrivastav, H.; Hiltz, S.R. Information overload in technology-based education: A meta-analysis. In Proceedings of the Nineteenth Americas Conference on Information Systems, Chicago, IL, USA, 15–17 August 2013.
29. Millar, R. *Beyond 2000: Science Education for the Future*; Nuffield Foundation: London, UK, 1998.
30. Nentwig, P.; Waddington, D. *Making It Relevant: Context Based Learning of Science*; Waxmann Verlag: Münster, Germany, 2006.
31. Hazeyama, A.; Shimizu, H. A learning environment for software security education. In Proceedings of the 2011 Fifth International Conference on Secure Software Integration and Reliability Improvement-Companion, Jeju Island, Republic of Korea, 27–29 June; IEEE: Piscataway, NJ, USA, 2011; pp. 7–8.
32. Hazeyama, A.; Shimizu, H. Development of a Software Security Learning Environment. In Proceedings of the 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Kyoto, Japan, 8–10 August 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 518–523.
33. Whitney, M.; Lipford-Richter, H.; Chu, B.; Zhu, J. Embedding secure coding instruction into the IDE: A field study in an advanced CS course. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, MO, USA, 4–7 March 2015; ACM: New York, NY, USA, 2015; pp. 60–65.
34. Zhu, J.; Lipford, H.R.; Chu, B. Interactive support for secure programming education. In Proceedings of the 44th ACM Technical Symposium on Computer Science Education, Denver, CO, USA, 6 March 2013; ACM: New York, NY, USA, 2013; pp. 687–692.
35. Schweitzer, D.; Brown, W. Using visualization to teach security. *J. Comput. Sci. Coll.* **2009**, *24*, 143–150.
36. Yuan, X.; Vega, P.; Qadah, Y.; Archer, R.; Yu, H.; Xu, J. Visualization tools for teaching computer security. *ACM Trans. Comput. Educ. TOCE* **2010**, *9*, 20. [CrossRef]
37. Bishop, M.; Dai, J.; Dark, M.; Ngambeki, I.; Nico, P.; Zhu, M. Evaluating secure programming knowledge. In Proceedings of the IFIP World Conference on Information Security Education, Rome, Italy, 29–31 May 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 51–62.
38. Irvine, C.E.; Thompson, M.F.; Allen, K. CyberCIEGE: Gaming for information assurance. *IEEE Secur. Priv.* **2005**, *3*, 61–64. [CrossRef]
39. Nerbråten, Ø.; Røstad, L. Hacmegame: A tool for teaching software security. In Proceedings of the 2009 International Conference on Availability, Reliability and Security, Jukuoka, Japan, 16–19 March 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 811–816.
40. Peffers, K.; Tuunanen, T.; Niehaves, B. *Design Science Research Genres: Introduction to the Special Issue on Exemplars and Criteria for Applicable Design Science Research*; Taylor & Francis: Abingdon, UK, 2018.

41. Winter, R. Design science research in Europe. *Eur. J. Inf. Syst.* **2008**, *17*, 470–475. [CrossRef]
42. Peffers, K.; Tuunanen, T.; Rothenberger, M.A.; Chatterjee, S. A design science research methodology for information systems research. *J. Manag. Inf. Syst.* **2007**, *24*, 45–77. [CrossRef]
43. Kuechler, W.; Vaishnavi, V.K.; Petter, S. The aggregate general design cycle as a perspective on the evolution of computing communities of interest. In *Design Science Research Methods and Patterns*; Auerbach Publications: Boca Raton, FL, USA, 2007.
44. Hevner, A.R.; March, S.T.; Park, J.; Ram, S. Design science in information systems research. *MIS Q.* **2004**, *28*, 75–105. [CrossRef]
45. Rivet, A.E.; Krajcik, J.S. Contextualizing instruction: Leveraging students' prior knowledge and experiences to foster understanding of middle school science. *J. Res. Sci. Teach. Off. J. Natl. Assoc. Res. Sci. Teach.* **2008**, *45*, 79–100. [CrossRef]
46. Kamina, P.; Iyer, N.N. From concrete to abstract: Teaching for transfer of learning when using manipulatives. In Proceedings of the Northeastern Educational Research Association (NERA), Rocky Hill, CT, USA, 21–23 October 2009.
47. Johannesson, P.; Perjons, E. *An Introduction to Design Science*; Springer: Berlin/Heidelberg, Germany, 2014.
48. Wand, Y.; Storey, V.C.; Weber, R. An ontological analysis of the relationship construct in conceptual modeling. *ACM Trans. Database Syst. TODS* **1999**, *24*, 494–528. [CrossRef]
49. Tudorache, T.; Nyulas, C.; Noy, N.F.; Musen, M.A. WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semant. Web* **2013**, *4*, 89–99. [CrossRef] [PubMed]
50. Wen, S.-F.; Katt, B. Managing Software Security Knowledge in Context: An Ontology Based Approach. *Information* **2019**, *10*, 216. [CrossRef]
51. W3C. SPARQL 1.1 Query Language. 2013. Available online: https://www.w3.org/TR/sparql11-query/ (accessed on 3 June 2022).
52. Brank, J.; Grobelnik, M.; Mladenic, D. A survey of ontology evaluation techniques. In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005), Ljubljana, Slovenia, 17 October 2005; pp. 166–170.
53. Hlomani, H.; Stacey, D. Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey. *Semant. Web J.* **2014**, *1*, 1–11.
54. NorthBridge, BlackDuck. 2016 Future of Open Source Survey. 2016. Available online: http://www.slideshare.net/blackducksoftware/2016-future-of-open-source-survey-results (accessed on 23 November 2022).
55. BlackDuck Software. 2017 Open Source Security and Risk Analysis. 2017. Available online: https://www.gcomtw.com/mailshot/Synopsys/2002BlackDuck/repossra19.pdf (accessed on 13 September 2018).
56. Snyk. The State of Open Source Security—2019. 2019. Available online: https://snyk.io/opensourcesecurity-2019/ (accessed on 2 November 2022).
57. Numally, J.C. *Psychometric Theory*; McGraw-Hill: New York, NY, USA, 1978.
58. Seel, N.M.; Al-Diban, S.; Blumschein, P. Mental Models & Instructional Planning. In *Integrated and Holistic Perspectives on Learning, Instruction and Technology*; Springer: Berlin/Heidelberg, Germany, 2000.
59. Turpin, K. OWASP Secure Coding Practices-Quick Reference Guide. 2010. Available online: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/ (accessed on 3 November 2022).
60. Nasehi, S.M.; Sillito, J.; Maurer, F.; Burns, C. What makes a good code example?: A study of programming Q&A in StackOverflow. In Proceedings of the 2012 28th IEEE International Conference on Software Maintenance (ICSM), Trento, Italy, 23–28 September 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 25–34.
61. Wen, S.-F. Learning secure programming in open source software communities: A socio-technical view. In Proceedings of the 6th International Conference on Information and Education Technology, Osaka, Japan, 6–8 January 2018; ACM: New York, NY, USA, 2018; pp. 25–32.
62. Nunamaker, J.F., Jr.; Briggs, R.O. Toward a broader vision for information systems. *ACM Trans. Manag. Inf. Syst. TMIS* **2011**, *2*, 20. [CrossRef]
63. Bider, I.; Kowalski, S. A framework for synchronizing human behavior, processes and support systems using a socio-technical approach. In *Enterprise, Business-Process and Information Systems Modeling*; Springer: Berlin/Heidelberg, Germany, 2014.
64. Kowalski, S. *IT Insecurity: A Multi-Discipline Inquiry*; Department of Computer and System Sciences, University of Stockholm and Royal Institute of Technology: Stockholm, Sweden, 1994.
65. Al Sabbagh, B.; Kowalski, S. A socio-technical framework for threat modeling a software supply chain. In Proceedings of the 2013 Dewald Roode Workshop on Information Systems Security Research, Niagara Falls, NY, USA, 4–5 October 2013; International Federation for Information Processing: Laxenburg, Austria, 2013.