

IFIP AICT 557

Lynette Drevin
Marianthi Theocharidou
(Eds.)



Information Security Education

Education in Proactive Information Security

12th IFIP WG 11.8 World Conference, WISE 12
Lisbon, Portugal, June 25–27, 2019, Proceedings

 Springer

Lynette Drevin · Marianthi Theocharidou (Eds.)

Information Security Education


Education in Proactive Information Security

12th IFIP WG 11.8 World Conference, WISE 12
Lisbon, Portugal, June 25–27, 2019
Proceedings

 Springer

Editors

Lynette Drevin 
North-West University
Potchefstroom, South Africa

Marianthi Theocharidou 
European Commission
Joint Research Center
Ispra, Italy

ISSN 1868-4238 ISSN 1868-422X (electronic)
IFIP Advances in Information and Communication Technology
ISBN 978-3-030-23450-8 ISBN 978-3-030-23451-5 (eBook)
<https://doi.org/10.1007/978-3-030-23451-5>

© IFIP International Federation for Information Processing 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Innovation in Curricula

An Educational Intervention for Teaching Secure Coding Practices	3
<i>Vuyolwethu Mdunyelwa, Lynn Fitcher, and Johan van Niekerk</i>	
Learning Principles and the Secure Programming Clinic.	16
<i>Matt Bishop, Melissa Dark, Lynn Fitcher, Johan Van Niekerk, Ida Ngambeki, Somdutta Bose, and Minghua Zhu</i>	
Introducing Research into the Undergraduate Curriculum in Cybersecurity . . .	30
<i>Dimitrios Damopoulos and Susanne Wetzel</i>	

Training

A Short-Cycle Framework Approach to Integrating Psychometric Feedback and Data Analytics to Rapid Cyber Defense.	45
<i>Erik L. Moore, Steven P. Fulton, Roberta A. Mancuso, Tristen K. Amador, and Daniel M. Likarish</i>	
Identifying Security Requirements Body of Knowledge for the Security Systems Engineer	59
<i>Suné von Solms and Annlizé Marnewick</i>	
Andragogy as a Scientific Basis for Training Professionals in Information Security	72
<i>Alexander Tolstoy and Natalia Miloslavskaya</i>	

Applications and Cryptography

Light Cryptography	89
<i>Pascal Lafourcade, Takaaki Mizuki, Atsuki Nagao, and Kazumasa Shinagawa</i>	
Blockchain and Its Security: Ignore or Insert into Academic Training?.	102
<i>Natalia Miloslavskaya and Alexander Tolstoy</i>	
Identifying Information Security Risks in a Social Network Using Self-organising Maps.	114
<i>Rudi Serfontein, Hennie Kruger, and Lynette Drevin</i>	

Organisational Aspects

Lessons Learned from an Organizational Information Security Awareness Campaign 129
Juan-Marc Scrimgeour and Jacques Ophoff

A Comprehensive Framework for Understanding Security Culture in Organizations 143
Alaa Tolah, Steven M. Furnell, and Maria Papadaki


Using Gamification to Improve Information Security Behavior: A Password Strength Experiment 157
Jacques Ophoff and Frauke Dietz

Author Index 171

Innovation in Curricula



An Educational Intervention for Teaching Secure Coding Practices

Vuyolwethu Mdunyelwa¹(✉) , Lynn Futcher¹ , and Johan van Niekerk^{1,2} 

¹ Nelson Mandela University, Port Elizabeth, South Africa
{vuyolwethu.mdunyelwa,lynn.futcher,johan.vanniekerk}@mandela.ac.za

² Noroff University College, Kristiansand, Norway
johan.vanniekerk@noroff.no

Abstract. Cybersecurity vulnerabilities are typically addressed through the implementation of various cybersecurity controls. These controls can be operational, technical or physical in nature. The focus of this paper is on technical controls with a specific focus on securing web applications. The secure coding practices used in this research are based on OWASP. An initial investigation found that there was a general lack of adherence to these secure coding practices by third year software development students doing their capstone project at a South African University. This research therefore focused on addressing this problem by developing an educational intervention to teach secure coding practices, specifically focusing on the data access layer of web applications developed in the .NET environment. Pre-tests and post-tests were conducted in order to determine the effectiveness of the intervention. Results indicated an increase in both knowledge and behaviour regarding the identified secure coding practices after exposure to the intervention.

Keywords: Educational intervention · Secure coding practices · OWASP · Web application security

1 Introduction

With the recent increase in cyber-related attacks, cybersecurity is becoming a key area of concern for many organisations. Web applications often handle very sensitive data, used for carrying out critical tasks such as banking, online shopping and online tax filing [9]. These applications are trusted by billions of users for performing such daily activities. However, 75% of all attacks on the internet are executed through the application layer of the OSI model [6], and more than 76% of web applications have vulnerabilities [2].

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the authors, and are not necessarily to be attributed to the NRF.

© IFIP International Federation for Information Processing 2019

Published by Springer Nature Switzerland AG 2019

L. Drevin and M. Theocharidou (Eds.): WISE 2019, IFIP AICT 557, pp. 3–15, 2019.

https://doi.org/10.1007/978-3-030-23451-5_1

Handling risks related to the security of web applications is a major challenge for many organizations. Not surprisingly, Web applications have recently received attention from academia and industry to initiate some defence mechanisms to protect them from security threats [9]. Many of these Web applications have common vulnerabilities which can be easily corrected [18] through introducing secure coding practices.

The secure coding practices used in this research are based on OWASP. An initial investigation found that there was a general lack of adherence to these secure coding practices by third year software development students doing their capstone project at a South African University. This research therefore focused on addressing this problem by developing an educational intervention to teach secure coding practices, specifically focusing on the data access layer of web applications developed in the .NET environment.

The following section highlights the related literature, while Sect. 3 provides the research design. Section 4 presents the educational intervention including both the knowledge and behavioural components. This is followed by Sect. 5 which provides the results of the verification of the educational intervention before concluding in Sect. 6.

2 Related Literature

More than 90,000 vulnerabilities have been recorded in the Symantec comprehensive vulnerability database over the past two decades, from 24,560 vendors representing over 78,900 products. On average, over 340,000 web attacks were blocked from web applications per day in 2014 [2]. Although this improved to 229,000 in 2016 [2], it still remains a serious concern since most attacks are no longer on the networks, but more on the software applications that run on the application layer. If 76% of web applications contain known vulnerabilities, it means that 24% of the scanned web applications do not contain known vulnerabilities. Therefore, it is possible for web applications to avoid known vulnerabilities. Those web applications without known vulnerabilities probably adhere to some form of best practice for secure software development. This is true as some researchers suggest that applying such practices and methodologies can improve security in software application [1, 7].

There are various organisations and institutions responsible for developing standards and best practices. These include the National Institute of Standards and Technology (NIST), the International Organizations for Standardization (ISO) and the International Electro-Technical Commission (IEC), the Microsoft Developer Network (MSDN) and the Open Web Application Project (OWASP) which provides best practices for improving security in web applications.

The best practices provided by these organisations were evaluated and OWASP was considered the most relevant for identifying fundamental secure coding practices to be taught to software developers. OWASP is known by many organisations for its Top 10 Vulnerability List (Table 1) that it publishes and updates periodically [4, 6, 11]. This list focusses on identifying the most serious

Table 1. OWASP top 10 vulnerability list 2017 [15].

Vulnerability	Description
SQL injection	Injection flaws occur when untrusted data is sent to an interpreter as part of a command or query
Broken authentication	This relates to authentication and session management that are often implemented incorrectly. It allows attacks to compromise information, and session tokens or exploit other implementation flaws
Sensitive data exposure	Many web applications and Application Programming Interfaces (APIs) do not properly protect data allowing attackers to steal or modify data
XML external entities	External entities can be used to disclose files using the file Uniform Resource Identifiers (URIs) handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks
Broken access control	Restrictions on what authenticated users are allowed to do are often not properly enforced, allowing attackers to exploit flaws to access unauthorized functionality, or data, such as other users' accounts or change access rights
Secure misconfiguration	This is mostly a result of insecure default configurations, incomplete or ad hoc configurations
Cross-site scripting	Also know as XSS, it allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect users to malicious sites
Insecure deserialisation	Insecure deserialisation leads to remote code execution, deserialisation flaws can also be used to perform tasks such as injection attacks, and privilege escalation attacks
Using components with known vulnerabilities	Components including libraries, frameworks and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such attacks can facilitate serious data loss or server take over
Insufficient logging and monitoring	This allows attacks to further attack systems, maintain persistence, pivot to more systems, and temper, extract, or destroy data

web application security vulnerabilities for many organisations [16]. The Top 10 list changes according to which vulnerability is most dominant at any given time.

The risk posed by each of these vulnerabilities can be reduced by more than one type of control. For the data access layer within .NET, OWASP recommends specific secure coding practices. Table 2 presents the nine secure coding practices (SP1 to SP9) for data access based on OWASP and used in this study. These relate to some of the vulnerabilities shown in Table 1. As an example, parameterised SQL commands (SP1), or the use of stored procedures (SP6), can block SQL injections. Therefore more than one control can reduce a vulnerability.

The secure coding practices shown in Table 2 are referred to using the codes SP1 to SP9 throughout this paper. If one of them is not properly handled, it

Table 2. Secure coding practices. Adapted from [16].

SP	Secure coding practices
SP1	Use parameterised SQL commands for all data access, without exception
SP2	Do not use SQL command with a string made up of a concatenated SQL strings
SP3	Properly validate input fields
SP4	Apply the principle of least privilege when setting up the database of your choice
SP5	When using SQL server, prefer integrated authentication over SQL authentication
SP6	Using stored procedures is the most effective way to counter the SQL injection vulnerability
SP7	Encrypt connection strings
SP8	Connection strings should be based in a configuration file
SP9	Encrypt sensitive data using acceptable encryption algorithms

can be easy for an attacker to access and modify information that is in the database. For example, if the connection string is found in other parts of the application code and not locked in the configuration file, it can be easy for an attacker to access the information using the same connection string to connect to the database. Or, if the expected values in an input field are not whitelisted in a system with concatenated SQL strings, attackers can use characters to manipulate the SQL string in the database and the information would be at risk.

These vulnerabilities cannot be prevented by programmers unless they know the types of flaws that exist in their code [1,3]. Similarly, they cannot implement these security controls unless they are taught how they work [8]. Once software developers have been taught about secure coding practices, it is more likely that they will have the requisite knowledge [5]. However, there has to be some form of compliance instrument to monitor their adherence, since it is known that people with the requisite knowledge do not always behave accordingly. Therefore, an educational intervention that focuses on both knowledge and behaviour was developed and provided to software development students to improve the security of their web applications.

3 Research Design

This research was conducted in the School of Information and Communication Technology at a comprehensive institution in South Africa, offering both degrees and vocational qualifications. In this case, the sample was drawn from students registered for their third year in the National Diploma: Software Development. In South Africa, there are no locally recognised curricular guidelines for computing. Many universities therefore rely on the recommendations provided in global

computing curricular publications. The Association of Computing Machinery (ACM) Information Technology curricular guidelines have been used to model IT qualifications. The IT2008 and the more recent IT2017 curricular guidelines require students in computing and engineering disciplines to engage in a capstone project during their final year of study [12, 13]. Since the diploma is a three year qualification, students are required to complete a capstone project in their third year of study. These capstone projects take place over a full year of study. According to the ACM IT curricular guidelines, capstone projects should typically adhere to the following [12, 13]:

- Project groups of 3 to 5 students;
- Based on a real-world problem;
- Must be integrative;
- Students should have completed most of the curriculum before attempting the project.

Students registered for the diploma are introduced to programming and business application systems development. Therefore, most of their capstone projects focus on developing applications for solving real world problems using business applications. When students choose the capstone projects, many of them focus on web, mobile or gaming applications, while a few develop desktop applications. Although students are taught specifically to develop software in a Windows environment using the .NET framework, students may develop their capstone projects in the programming language of their choice. Most project students choose web applications in the .NET development environment as this is where their skills lie.

This research focused on two aspects relating to secure coding practices, namely knowledge and behavioural compliance of the students and involved four main phases:

- **Phase 1** was the first phase for this research which started off by analysing students' behaviour relating to secure coding practices. This was done by conducting a code review on previously completed third year capstone projects, which were developed in the .NET environment. The results for this behavioural analysis indicated low levels of compliance to the identified secure coding practices.
- **Phase 2** addressed the knowledge assessment phase for this research, which assessed students' knowledge relating to secure coding practices. This was achieved using a questionnaire, which served as a pre-test for this study. Results from the pre-test indicated that students lacked knowledge relating to secure coding practices. Therefore, students lacked in both the knowledge and behavioural aspects.
- **Phase 3** comprised of an educational intervention for addressing both the knowledge and behavioural aspects, which students lacked in Phase 1 and 2. In terms of the knowledge aspect, students were provided with online lessons relating to secure coding practices to work through; in terms of the behavioural aspect, students were given a checklist to check their application code against the listed secure coding practices.

- **Phase 4** involved the verification of the educational intervention for this research. The first part of this phase was the knowledge verification (Phase 4A), and the second part was the behavioural verification (Phase 4B).

The results for Phase 1 and 2 were published in the 2017 Human Aspects in Information Security and Assurance (HAISA) conference [14]. The focus of this paper is therefore on Phases 3 and 4. The following section describes the educational intervention (Phase 3), while Phase 4 (A and B) are discussed in Sect. 5.

4 Educational Intervention

The educational intervention was split into two parts, where the first part focused on the knowledge, and the second part focused on the behaviour of students relating to secure coding practices. Owing to the lack of knowledge on the part of the students, the researcher realised the need to create a knowledge component that could assist students in acquiring the requisite knowledge regarding secure coding practices. The need to address behavioural compliance was also realised since it is known that having knowledge does not necessarily ensure that people would behave accordingly [17]. Both the knowledge and behavioural components of this research were designed using the identified secure coding practices in Table 2.

4.1 Knowledge Component

The knowledge component for this research took the form of a blended learning course, called the Web Application Security Course, that students worked through to improve their knowledge regarding secure coding practices.

Design of the Knowledge Component. The knowledge component for this research included online lessons that the researcher designed using the identified secure coding practices. For each of the secure coding practices, their importance and the security implications if they were ignored were explained. The online learning platform that was used to design the lessons was the *Moodle Learning Management System* that runs on the university’s website. Moodle is a learning management system used by educators to create effective blended learning material for students in various higher educational institutions. Moodle has been adopted by many institutions for its cost effectiveness, its ability to expand with increased student populations, and its ability to meet the needs of institutions, students and educators [10]. Figure 1 provides an overview of the process followed by the students when completing the online lessons on Moodle.

The lessons took the form of interactive Microsoft PowerPoint slides, which were converted to videos, for students to work through. Each secure coding practice was addressed in a single lesson. After completing each lesson, the students were required to take a quiz, which allowed them to reflect on the content of the

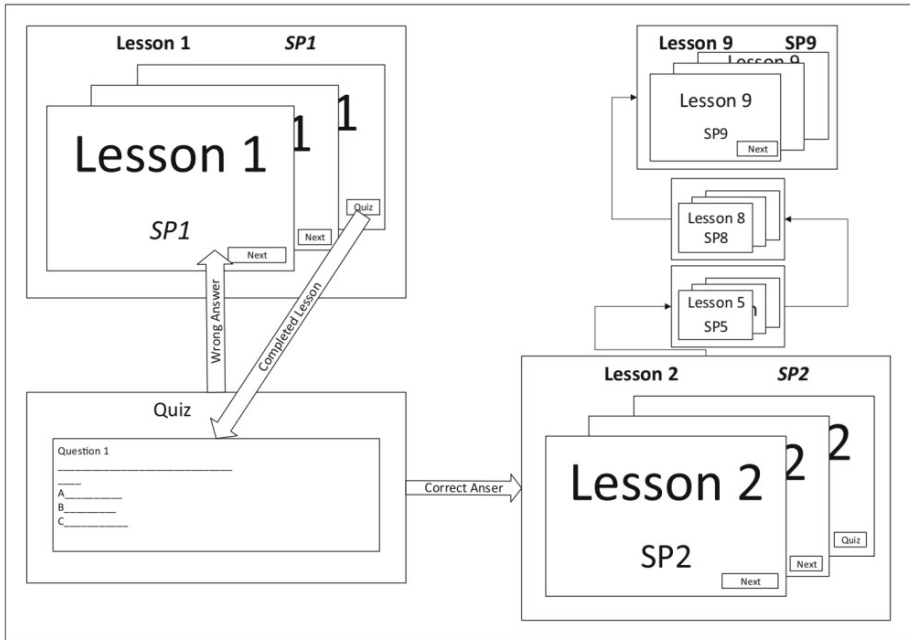


Fig. 1. Lesson content process flow.

lesson. The quiz had four questions assigned to each lesson. The students had to answer only one randomly generated quiz question before continuing to the next lesson. If the student selected the incorrect answer, they were required to work through the lesson again, and if they selected an incorrect answer once again, a different question would be randomly generated. Alternatively, if they selected the correct answer, they were allowed to continue to the next lesson.

A brief overview of each secure coding practice (SP1 to SP9) as listed in Table 2, within the knowledge component follows:

- SP1 (Using Parameterised SQL commands): The content for this secure coding practice firstly provides the background relating to parameterised SQL commands in order to equip students with the necessary information relating to this secure coding practice. The remainder of the lesson shows the students how parameterised SQL commands can be implemented in their code, and why it is necessary to use them.
- SP2 (Concatenated SQL strings): Content for this secure coding practice begins by introducing what is meant by concatenated SQL strings. The lesson proceeds by showing how programmers make use of concatenated SQL strings and the negative implications of using them. This lesson also provides a way in which to avoid using concatenated SQL strings, which is by means of parameterised SQL commands.

- SP3 (Input validation): The content for this secure coding practice begins by discussing validation in general. It also highlights the various types of validation, such as blacklisting and whitelisting, and why they are important. The content also provides suggestions on what to use when dealing with validation, for example, ASP .NET Regular Expressions to tell input fields which values to accept.
- SP4 (Principle of Least Privilege): This secure coding practice content explains what the Principle of Least Privilege is and why it is important when developing web applications. This content also provides a scenario where the use of this secure coding practice is shown and how it can be implemented.
- SP5 (Authentication): The content of this secure coding practice was addressed by means of a video adapted from YouTube. The video was embedded in the slides and distributed as a single lesson to the students to listen to and to watch.
- SP6 (Using Stored Procedures): The content for this secure coding practice focusses on how stored procedures are used and why they are important, providing examples on how they should be implemented in a web application.
- SP7 + SP8 (Connection strings): These two secure coding practices both deal with connection strings, and have been addressed collectively in the same lesson. The content first provides detail about the importance of connection strings, and how they should be handled when developing web applications, providing detail on how to implement both the secure coding practices.
- SP9 (Encryption): For this secure coding practice, an analogy is used to explain the concept of encryption to the students. The lesson further explains the analogy to clarify the concepts for the students. Since OWASP provides recommendations relating to acceptable encryption algorithms, the content for this lesson also emphasises the use of the encryption algorithms recommended by OWASP when developing web applications.

All the lessons were followed by a quiz question to check students' understanding of the content contained in the lesson they had worked on. The results for the short content quizzes were not recorded, since answers were simply used to ensure that students do not move to the next lesson without understanding the content in the previous lesson.

Administering the Knowledge Component. The Web Application Security lessons were prepared by the researcher and distributed to the students on Moodle. The students were permitted to work through the lessons as often as they wanted. During a lecture, the researcher explained the process that the students needed to follow when completing the online content. Most students worked through the content in the computer laboratories at the university as soon as it was made active and available to them. A total of 120 students completed the online lessons. The students had to work through all the lessons, since they were required to take a quiz which served as a Post-test (Phase 4A) for which marks were recorded.

4.2 Behavioural Compliance Monitoring Instrument

Although it is possible for a student to have the requisite knowledge and not perform accordingly when developing their web applications, it is most unlikely for them to behave accordingly when they do not have the requisite knowledge. Therefore, it was deemed necessary to firstly educate the students on secure coding practices and then to monitor their adherence to these practices. This section provides details on how the behaviour of students was monitored when developing their web applications as part of their third year capstone projects.

Design of Behavioural Compliance Instrument. The behavioural compliance instrument took the form of a checklist as seen in Table 3. The code review checklist for this research was adapted from the secure coding practices in Table 2 and was provided to the students electronically via Moodle.

Table 3. Code review checklist.

SP	Questions
SP1	Do they make use of parameterised SQL commands for all data access? <i>Yes/No (Number of Instances)</i>
SP2	Do they make use of concatenated strings in the queries? <i>Yes (Number of Instances)/No</i>
SP3	Are all input fields validated? <i>Input Properly Validated/Input not Properly Validated/No Validation</i>
SP4	Do they make use of the Principle of Least Privilege when setting up their databases? <i>Yes/No</i>
SP5	Do they use integrated authentication or do they use SQL authentication? <i>Integrated Authentication/SQL Authentication</i>
SP6	Do they use stored procedures for their queries? <i>Yes/No/Inconsistent Use of Stored Procedures and Queries</i>
SP7	Do they encrypt their connection strings? <i>Yes/No</i>
SP8	Does the connection string only appear once in the web.config file? <i>Yes/No</i>
SP9	Is all the sensitive data being encrypted using the OWASP recommended methods? <i>Encrypted Using Acceptable Method/No Encryption/Encrypted Not using Acceptable Method</i>

Conducting the Behavioural Compliance Instrument. During a lecture the researcher explained to the students how they should go about using the checklist to review their capstone projects. They were required to check all web forms accessing the data access layer against the secure coding practices for SP1 to SP9 using the checklist provided in Table 3. Having worked through the knowledge component, as discussed in Sect. 4.1, the students should have

acquired the requisite knowledge relating to the secure coding practices that should be implemented in their web applications.

Since most students worked in groups when developing their web applications, they were also required to conduct a peer code review on each other's web forms using the checklist provided. The peer code review helped the students to double check whether they had really adhered to the secure coding practices as indicated in their own code reviews. Feedback from the students was positive and most students found the checklist helpful for their code and to ensure compliance to the secure coding practices.

5 Effectiveness of the Educational Intervention

Once students had completed the educational intervention, it was necessary to determine its effectiveness. The knowledge component of the educational intervention was responsible for providing students with knowledge regarding secure coding practices. Having completed the online course, the students were expected to implement the learnt secure coding practices in their capstone projects, showing behavioural compliance.

The verification of the knowledge component was achieved through an online quiz distributed to the students through the Moodle site as discussed in Sect. 5.1. Verification of the behavioural compliance component took the form of a code review by the researcher on the students' capstone projects as discussed in Sect. 5.2.

5.1 Knowledge Verification

The setup for the post-test questionnaire was such that students were only allowed to work through the post-test after they had completed the lessons in the knowledge component of the educational intervention, referred to as the Web Application Security Course. The 113 students who completed the post-test were only allowed to work through the post-test once. The results for the post-test questionnaire were automatically recorded on the Moodle site, where the researcher was able to export the data to an Excel spreadsheet for analysis. A comparison of the knowledge pre- and post-test is shown in Table 4.

Table 4. Knowledge assessment and verification results (Pre-test vs Post-test).

Phases	SP1	SP2	SP3	SP4	SP5	SP6	SP7	SP8	SP9
Phase 2	74%	36%	30%	58%	26%	39%	20%	3%	1%
Phase 4A	95%	95%	89%	91%	91%	77%	93%	88%	83%
Variance	21%	59%	59%	33%	65%	38%	73%	85%	82%

Table 4 shows the results for Phase 2, Knowledge Assessment (pre-test), and Phase 4A, Knowledge Verification (post-test). There was a substantial improvement in the students' knowledge as indicated in the second row, Phase 4A. Students' knowledge has improved in all of the secure coding practices (SP1 to SP9), as seen in the variances. SP2, SP3, SP5 and SP7 showed reasonable improvements, while SP8 and SP9 showed the highest improvements with variances above 80%. As mentioned previously, knowledge acquisition does not guarantee a change in behaviour. In order to monitor the adherence of the students to these secure coding practices when developing their web applications behavioural compliance monitoring was required.

5.2 Behavioural Verification

The behavioural verification instrument used was the same checklist used in Phases 1 and 3 as shown in Table 3. The checklist was used by the researcher to conduct a code review on the third year capstone projects.

The code review was conducted by the researcher before the final submission of the software development projects. The researcher first informed the students about the code review process scheduled to take place during a session in the computer laboratory. Students filled in their group names and were required to be in the computer laboratory in order for their projects to be reviewed. The code review was conducted during the students' practical sessions. For each of the capstone projects, the researcher reviewed five web forms per project, which connected to the database and were related to the capstone projects' main functionality. 17 groups were present for the code review, and they were all reviewed successfully, in the presence of the students who belonged to the group being reviewed. Table 5 shows the results from the behavioural analysis for the students before and after exposure to the educational intervention.

Table 5. Behavioural verification results (Phase 1 and 4B).

Phases	SP1	SP2	SP3	SP4	SP5	SP6	SP7	SP8	SP9
Phase 1	86%	84%	77%	60%	N/A	38%	N/A	68%	31%
Phase 4B	96%	96%	100%	91%	N/A	96%	N/A	100%	100%
Variance	10%	12%	23%	31%	N/A	58%	N/A	32%	69%

As can be seen from the results in Table 5, there was an improvement in students' adherence to secure coding practices after the educational intervention, with most capstone project groups having adhered to all the secure coding practices. Although SP5 and SP7 were recommended by OWASP, they were not required by the capstone projects from which the sample for this research was drawn. All averages per secure coding practice were between 90% and 100%, with SP3, SP8, and SP9 showing 100% compliance. SP6 and SP9 showed the

largest improvements of 58%(SP6) and 69%(SP9) respectively, while SP3, SP4 and SP8 showed good improvements of between 20% and 35%.

6 Conclusion

The results of this study indicate that students' adherence to secure coding practices can be positively impacted through a formal educational intervention. However, it is important that such an intervention addresses both the knowledge and behaviour of students since having the requisite knowledge does not ensure compliance. It is for this reason that a behavioural compliance monitoring instrument formed part of the study. This is a step towards educating students in secure application development which is essential in addressing the many security vulnerabilities existing in Web applications today.

Limitations of this study do exist. Firstly, this study addressed only the identified secure coding practices which were determined from OWASP. Secondly, the identified secure coding practices only focused on the data access layer of Web applications developed in the .NET environment. Future research could investigate similar interventions within various other application development contexts.

7 Ethical Considerations

This research project adhered to all ethical requirements of the Nelson Mandela University and obtained ethics approval from the university research committee (REF H15-ENG-ITE-009).




References

1. Bishop, M., Dai, J., Dark, M., Ngambeki, I., Nico, P., Zhu, M.: Evaluating secure programming knowledge. In: Bishop, M., Fitcher, L., Miloslavskaya, N., Theocharidou, M. (eds.) WISE 2017. IAICT, vol. 503, pp. 51–62. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58553-6_5
2. Chandrasekar, K., Cleary, G., Cox, O., Gorman, B.O.: Internet security threat report. Technical report, Symantec, April 2017. <https://www.symantec.com/security-center/threat-report>
3. Chi, H., Jones, E.L., Brown, J.: Teaching secure coding practices to STEM students. In: Proceedings of the 2013 on InfoSecCD 2013 Information Security Curriculum Development Conference - InfoSecCD 2013, pp. 42–48 (2013). <https://doi.org/10.1145/2528908.2528911>. <http://dl.acm.org/citation.cfm?doid=2528908.2528911>
4. Chung, S., et al.: What approaches work best for teaching secure coding practices? In: 2014 HUIE Education & STEM Conference (2014)
5. Conklin, A., White, G.: A graduate level assessment course: a model for safe vulnerability assessments. In: Proceedings of the 9th Colloquium for Information Systems Security Education (2005)

6. Customs Solutions Group: A CISO's guide to application security. Technical report, Customs Solutions Group (2012). <http://h30528.www3.hp.com/Security/CISOGuideToApplicationSecurity.pdf>
7. Dark, M., Ngambeki, I., Bishop, M., Belcher, S.: Teach the hands, train the mind ... a secure programming clinic! In: Proceeding of the 19th Colloquium for Information System Security Education (2015)
8. Dark, M., Stuart, L., Ngambeki, I., Bishop, M.: Effect of the secure programming clinic on learners' secure programming practices. *J. Colloquium Inf. Syst. Secur. Educ.* **4**, 18–18 (2016)
9. Deepa, G., Thilagam, P.S.: Securing web applications from injection and logic vulnerabilities: approaches and challenges. *Inf. Softw. Technol.* **74**, 160–180 (2016). <https://doi.org/10.1016/j.infsof.2016.02.005>
10. Florian, T.P., Zimmerman, J.P.: Understanding by design, moodle, and blended learning: a secondary school case study. *MERLOT J. Online Learn. Teach.* **11**(1), 120–128 (2015)
11. Li, X., Xue, Y.: A survey on server-side approaches to securing web applications. *ACM Comput. Surv. (CSUR)* **46**(4), 54 (2014)
12. Lunt, B., Sabin, M., Hala, A., Impagliazzo, J., Zhang, M.: Information technology curricula 2017. Technical report, Association for Computing Machinery (ACM). IEEE Computer Society (2017)
13. Lunt, B.M., Ekstrom, J.J., Lawson, E.: Curriculum guidelines for undergraduate degree programs in information technology. Technical report, Association for Computing Machinery (ACM). IEEE Computer Society (2008)
14. Mdunyelwa, V.S., Niekerk, J.F.V., Futcher, L.A.: Secure coding practices in the software development capstone projects. In: Proceedings of the Eleventh International Symposium on Human Aspects of Information Security & Assurance, HAISA 2017, Adelaide, Australia, 28–30 November 2017, pp. 282–291 (2017). <http://www.cscan.org/openaccess/?paperid=353>
15. OWASP: OWASP Top 10 (2017). https://www.owasp.org/index.php/Top_10-2017_Top_10
16. OWASP: The OWASP Foundation (2017). https://www.owasp.org/index.php/Main_Page
17. Vroom, C., Von Solms, R.: Towards information security behavioural compliance. *Comput. Secur.* **23**(3), 191–198 (2004)
18. Zhu, J., Xie, J., Lipford, H.R., Chu, B.: Supporting secure programming in web applications through interactive static analysis. *J. Adv. Res.* **5**(4), 449–462 (2014). <https://doi.org/10.1016/j.jare.2013.11.006>



Learning Principles and the Secure Programming Clinic

Matt Bishop¹ , Melissa Dark², Lynn Fletcher³ , Johan Van Niekerk^{3,4} ,
Ida Ngambeki², Somdutta Bose¹, and Minghua Zhu¹

¹ University of California at Davis, Davis, CA, USA
{mabishop,sombose,mhzhu}@ucdavis.edu

² Purdue University, West Lafayette, IN, USA
{dark,ingambek}@purdue.edu

³ Center for Research in Information and Cyber Security,
Nelson Mandela University, Port Elizabeth, South Africa
{lynn.fletcher,johan.vanniekerk}@mandela.ac.za

⁴ Noroff University College, Oslo, Norway
johan.vanniekerk@noroff.no

Abstract. Several academic institutions have run a clinic on robust and secure programming. Each time a clinic was run, it was associated with a specific class. Using pre- and post-class evaluation instruments, it is clear that the effect of the secure programming clinic on students' understanding of secure programming was generally positive. However, in some instances the clinic was underutilized, and in other cases it could not be run at other institutions. The goal of this paper is to examine the structure of the clinic in light of five basic learning principles, and provide information about when a clinic will not improve students' understanding, and when it will. We validate this by examining an instance of the secure programming clinic, and show how the learning principles explain the improvement in student grades, or lack thereof. From this, we draw conclusions about ways to make the clinic more effective, and when it will not be effective.

Keywords: Secure programming clinic · Learning principles · Robust programming

1 Introduction

The problem of nonsecure code is widely recognized as a major source of security problems. Indeed, of the vulnerabilities in the U.S. National Vulnerability Database in the last 5 years, over 19,000 are identified as injection and buffer overflow vulnerabilities, exemplars of poor programming practices [14]. Some, such as Heartbleed, have impacts throughout the Internet [7]. Industries, government, and many other organizations want programmers who can write secure, robust code. The problem is how to teach this material.

© IFIP International Federation for Information Processing 2019

Published by Springer Nature Switzerland AG 2019

L. Drevin and M. Theocharidou (Eds.): WISE 2019, IFIP AICT 557, pp. 16–29, 2019.

https://doi.org/10.1007/978-3-030-23451-5_2

3. The clinicians can act as assistant instructors, helping the students develop threat models for how an attacker might use their program to violate desired security properties. As “security” is defined in terms of requirements, the threat model is critical to knowing the types of security problems that might arise. On the other hand, robustness issues are independent of threats, in the sense that they are common to all threats.

The functions of the clinic can be extended beyond simply reviewing programs. It can also provide information to help the students fix the problems. This typically requires collecting examples of poor programming and how to fix, or (better) avoid, them. It can also provide remote assistance, where the clinicians are not at the institutions. There is a salutary effect for this. If some of the clinicians are volunteers who work in the software industry or for government agencies, their presence and activities will convey the importance that future employers place in high-quality code. This provides incentives for students to learn the material.

The clinic can also be shared among universities. One implementation of the clinic provides a common shared appointment calendar, so students from any of the academic institutions could sign up for appointments even when the local clinicians were not available. The clinicians from the institutions coordinated their times so that one was always available during the day. Were this to be extended internationally, clinicians would probably be available for most of the evening and night (when many students of computer science and related disciplines develop their programs).

The above discussion provides insight into the specific secure programming clinic format of concern to this paper. However, for the sake of comprehensiveness, the next section will briefly highlight other such approaches, and challenges, relating to the teaching of secure programming.

4 Teaching Secure Programming

Secure programming is about writing secure code. The focus of many programming courses, however, is to write code that works with a lack of focus on writing code securely. A developer’s unintentional ignorance of known vulnerabilities and insecure coding practices can generate insecure software. Besides the potential financial loss, the successful exploitation of insecure software can impact the confidentiality, integrity and availability (CIA) of critical information. Undetected exploitation can also lead to the embedding of malicious software within an organization, giving the malicious attacker the ability and potential to attack any time [18]. Secure programming should therefore include the basic principles of robust coding to guard against unexpected inputs and events [15].

The challenges of teaching and integrating secure programming into computing curricula have been around for many years, and some of these challenges which are still evident today [13]. These include:

- Lack of faculty buy-in
- Competition with other topics for inclusion into the curriculum
- Computing curricula already full
- Failure of students to grasp other important programming concepts
- Lack of secure programming expertise of faculty members

Much research has been conducted to address some of these challenges. A recent study [21] investigates a Java proof-of-concept plug-in for Eclipse, ESIDE (Educational Security in the IDE), that provides vulnerability warnings and secure programming education in the IDE while students write code. It works by scanning a selected project for code patterns that match predefined heuristic rules of security vulnerabilities. In this way, secure programming knowledge can be introduced early and reinforced throughout a students' education. Generally, ESIDE was found to increase students' awareness and knowledge of secure programming. However, almost no students actually modified their code to mitigate the detected vulnerabilities as they were most concerned with completing functionality and did not want to impact that functionality with additional security-oriented code. In addition, carefully timing the introduction of concepts and skills as well as incentivising such learning is important [21].

ESIDE was compared to the Secure Programming Clinic by running each approach with two separate groups of students, one group assigned to ESIDE and the other to the clinic [21]. Each group of students were asked to report on how likely they would use the recommended changes in their code during the session. The likelihood results for the Secure Programming Clinic were significantly better than for ESIDE. However, the clearest difference between the clinic and ESIDE were the number of specific vulnerabilities covered. Where ESIDE marked on average 42 lines of code per participant, the technical assistants running the clinic pointed out approximately two specific lines of code per participant.

One response to the need to teach students to program more securely was to introduce a serious game for teaching secure coding practices and principles to novice programmers [1]. Initial findings showed the game to be usable and engaging, with the majority of students being able to make clear correlations between the game levels and corresponding security concepts. Similarly, constructing secure coding duels [24] in Code Hunt, a high-impact serious gaming platform released by Microsoft Research, was proposed to instill gaming aspects into the education and training of secure coding. Secure coding duels proposed in this work are coding duels that are carefully designed to train players' secure coding skills, such as sufficient input validation and access control. Using serious games for teaching secure coding could alleviate some of the challenges faced by faculty members in this regard.

Furthermore, scorecards and checklists provide a consistent means of evaluation and assessment [22]. They describe the use of security checklists and scorecards which provide a quantifiable list of security criteria to aid in writing secure code and further reinforce security principles. Checklists distributed to students included:

- Sample code of errors to look for;
- Examples of correct ways of writing code; and
- Security mantras including a list of principles that form the basis for the checklist, for example: “All Input is Evil!”

Regardless of the approach used to teach secure programming, such approaches should take into account recognized learning principles, as discussed in Sect. 5, to ensure that learning takes place.

5 Learning Principles

Systematic studies of human behaviour, including studies of how people learn, is a relatively new field of scientific enquiry [17]. However, despite the youth of this field, many studies have already been dedicated to investigating how learning takes place. During such studies, researchers strive to identify recurring patterns in the data and to make generalizations based on these patterns. Such generalizations lead to the formulation of *learning principles* and *learning theories*. Learning principles identify the factors that influences learning. For example, the *principle* that a behaviour which is rewarded in some way is more likely to re-occur in future than one which is not followed by a reward. A *learning theory* on the other hand aims to provide an explanation of the underlying mechanisms that are involved in learning. Thus, whilst a learning principle presents *what* factors are important, a learning theory would explain *why* those factors are important [17].

Learning *principles* do not change much over time, however, learning *theories* have continually changed as understanding of human behaviour evolved [17]. Due to the fact that learning principles are less changeable, and thus more ‘future proof’ than learning theories, this research will seek to identify learning *principles* that could be useful in the secure coding clinics, but will avoid subscribing to any specific learning *theory*.

Educational literature provides many such learning principles. These principles have been identified, and their impact verified, in a variety of ways. One such approach is the field of brain compatible education. This educational approach stems from a combination of neuroscience and educational psychology and was first made possible by advances in brain imaging during the 1990s [12].

Brain-compatible, or brain-based, learning is not a formalised education approach or ‘recipe for teachers’, instead it provides a “set of principles and a base of knowledge and skills upon which we can make better decisions about the learning process” [9, p. xiii]. Brain research has shown that humans literally grow new dendrites and neural connections every time they learn something. Knowing which educational activities are the most effective in stimulating such growth allows educational practitioners to create material that leverages the way the brain naturally learns [10]. For the purpose of this research, it is not necessary to understand how these natural learning processes work. One only needs to understand that these principles were verified as being effective in promoting real learning.

No single complete list of such principles exists. However, many principles are presented and discussed in the literature [4, 5, 8, 9, 11, 16, 19, 20]. The list presented in Table 1 contains a subset of principles from those used in literature. The principles included in Table 1 were restricted to those the authors specifically deemed most relevant to the context of the Secure Programming Clinic. Relevant principles from literature were reworded and consolidated in cases where there was significant overlap in meaning between those used in the literature and the context for use in this study. Table 1 thus presents the authors' adaption of these principles. The following discussion briefly elaborates on each of the listed principles:

- LP1 - According to [8, 9, 11] there is no long term retention without rehearsal. The brain would prune new neural growth if it is not reinforced by being used. It is vital to repeat lessons taught more than once, otherwise students would be likely to forget these lessons. One should also allow enough time for students to assimilate any new concepts. Several studies [4, 8, 11] explains that the brain will reconsolidate new neural growth for several weeks after learning using both conscious and unconscious (sleep) processes to decide how to incorporate knowledge into existing neural structures.
- LP2 - If the new knowledge is too advanced for the target audience, learning might be inhibited because the learners feel threatened instead of challenged by the content [4, 5, 8, 9]. Furthermore, new knowledge can only be assimilated if it builds upon prior knowledge, since novel patterns can only form as extensions of existing patterns [5, 11, 19, 20].
- LP3 - The process of learning consists of the brain recognizing patterns [4, 5, 8, 9]. For these patterns to form the learners need to recognize and connect patterns by themselves [5, 9, 11, 19]. This process works best if the learners experience these patterns in contexts that are relevant to themselves [5, 9] and their real-life experiences [11].
- LP4 - Humans naturally learn in social settings and through interaction with others [4, 8]. Collaboration with others enhances learning [11].
- LP5 - Rehearsal will make learning permanent, however, this does not guarantee the rehearsed learning is in fact correct. Practice should be accompanied by *feedback* that is constant, consistent, and specific to ensure that practice that is permanent is also correct [8, 16]. The effect of feedback is also amplified if it is immediate [5, 9].

6 Mapping of Clinics to Selected Learning Principles

We begin by examining the instances of the secure programming clinic that have been run, and how they reflect the learning principles. We then discuss how the clinic might be improved by mapping the principles into various forms of the clinic.

scripts in near real time. This came to be known among the group as the Short-Cycle Framework for psychometric coaching feedback.

The types of data gathered by the authors to enable the Short-Cycle Framework fall into baseline data (MBTI and PTPS traits), established before exercise scenarios and cyber defense incidents, and ongoing data (PTPS adaptive behaviors) that is gathered with different frequencies throughout the event. This all could then be plotted on the same timeline with traditional cybersecurity training metrics listed below and described in detail in later sections.

- Personality Trait Assessments
 - Myers-Briggs Type Indicator (MBTI)
 - Parker Team Player Survey (PTPS)
 - Adaptive Behavior Scale
 - 14-Item Resilience Scale
- Ongoing Data
 - Personality State Assessments
 - Team Cohesion Assessment Scale
 - Observed PTPS
 - Digital Observations through System and Event, Information Management (SIEM) Network and Log Traffic Data from devices like Firewalls, Server Logs, and Switch Flow Traffic
 - Digital Service Scoring Engine - tracking business digital system state over time of the defended services.
 - Red Team Journaling - presents the active attack and often intentionality
 - “CEO” Injects - provides the timelines of directives issued by a mock CEO

Psychometric state analysis like “Team Cohesion” during events makes ongoing relationship dynamics and causal events of behavior clearer. This is why data were aggregated into a single timeline. To meaningfully address this, the team is developing a set of scripted feedback messages for particular psychometric states that can be evaluated for efficacy as more event data are collected. More detailed explanations of

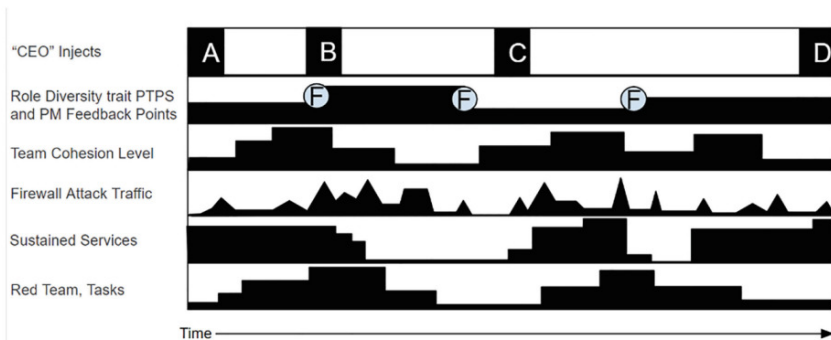


Fig. 1. Short-cycle framework, a simulated set of metrics across cyber defense and team psychometric indicators set in parallel on a timeline to rapidly analyze cyber event causes in relation to ongoing psychometric measurement of the team.