https://doi.org/10.48009/2_iis_2019_1-8

Issues in Information Systems
Volume 20, Issue 2, pp. 1-8, 2019

A COMPARATIVE STUDY ANALYZING COMPUTER PROGRAMMING COLLEGE STUDENTS' PRE-KNOWLEDGE AND POST-KNOWLEDGE OF SOFTWARE APPLICATION SECURITY USING OWASP

Mel Tomeo, Pittsburgh Technical College, tomeo.mel@ptcollege.edu
Wilfred Mutale, Pittsburgh Technical College, mutale.wilfred@ptcollege.edu
John Scarpino, Pittsburgh Technical College, scarpino.john@ptcollege.edu
Lee Cottrell, Pittsburgh Technical College, cottrell.lee@ptcollege.edu

ABSTRACT

Software application security has become increasingly important in a society that relies heavily on information systems and technology. Business organizations and consumers alike have suffered great losses as a result of compromised web applications. One of the steps towards reducing software application security vulnerabilities is to familiarize computer programming students to the Open Web Application Security Project (OWASP) proactive controls of secure coding best practices. In this study, faculty conducted a survey of 37 computer programming college students to evaluate their knowledge of software application security before and after being educated on the OWASP Top Ten proactive controls best practices. The data gathered in this study were collected between February 2019 and March 2019 through SurveyMonkey. The participants in the survey were college students pursuing an undergraduate degree in computer programming and were in their first year of the program.

Keywords: Information Security (IS), OWASP, Secure Coding, Software Security, Application Security, Vulnerability Detection, Code Performance, Computer Programming, Information Systems, Software Development

INTRODUCTION

Security of software application involves activities carried out from the requirements, design, implementation, testing, release, and maintenance phases of the Software Development Life Cycle (SDLC) (Kissel, Stine, Scholl, Rossman, Fahlsing, & Gulick, 2008). Organizations have a responsibility to protect sensitive data within applications. Therefore, critical data must be encrypted while it is at rest and in transit. This includes financial transactions, web data, browser data, and information residing in mobile apps (Seacord & Rafail, 2006). Hence, developers should be security-focused when developing software applications. The goal is to help software developers build more secure software and reduce security vulnerabilities in the design of the code (Howard, 2004). The most effective secure coding best practice is to conduct ongoing education or training for software development teams. Developers should not design, develop, test, and document secure systems until they know the issues (Howard, 2004). Education should not highlight developers' weaknesses, but should provide awareness and exemplifies what happens when code contains security vulnerabilities.

This study investigates undergraduate computer programming college students' pre-knowledge and post-knowledge of secure coding best practices as recommended by the Open Web Application Security Project (OWASP). The data generating instruments for determining the students' pre- knowledge and post-knowledge of secure coding practices were collected from the survey questionnaire deployed through SurveyMonkey, an online survey platform. The questionnaire for the survey evaluated students' knowledge pre (before) and post (after) being educated on the OWASP proactive controls. Security techniques should be applied at an early stage in the life cycle of a software development phase to ensure maximum effectiveness (Anton, Manico, & Bird, 2018). Secure coding ensures that the program is generating output and behavior as intended by the developer. Security vulnerabilities and defects are results of poorly-constructed software that can lead to easy exploitation by cyber criminals (Aljawarneh, Alawneh, & Jaradat, 2017). Software application security assists in identifying and managing possible risks to networks and programs (Chahar, Chauhan, & Das, 2012).

LITERATURE REVIEW

A developer can use many different approaches to improve the quality of software. One particular approach investigated by Bishop (2010) gave developers the knowledge and skills on how to create programs that can resist attacks and handle errors appropriately. Bishop (2010) defines secure programming as "a style of programming used to create programs that can't be compromised" (p. 54). A study by Whitney, Lipford, Chu, and Thomas (2018) theorized that students should be educated on secure coding across a computing curriculum. The researchers conducted an investigation that took traditional classroom instruction and modified it into an educational resource inside a student's integrated development environment (IDE). From a sample of 45 interaction logs from the IDE, the researchers found a significant increase in the participants' secure coding awareness (Whitney et al., 2018). Their results were similar to a previous study by Whitney (2015) which indicated that students became aware of their coding behavior and increased their knowledge in security awareness and secure programming.

According to a strategies report by Warren, Favole, Haber, and Hamilton (2016), cybercrime costs the global economy an average of about $450 billion each year. This same report indicated that 828 million individual data records have been breached and exposed worldwide. In a report by Accenture Finance and Risk Services, public companies globally will risk losing $5.2 trillion to cybercrimes this year (Culp, 2019). Damages from cybercrimes can cost companies not only money, but also sensitive information. Cybercrimes can ruin the reputation of a company and also disrupt business and shareholders' value. One example of a cybercrime that companies are losing money to yearly is software piracy (Richter, 2019). The Software and Information Industry Association (SIIA) defined software piracy as the practice of unauthorized duplication of computer software (Givon, Mahajan, & Muller, 1995). In the United States alone, software piracy costs corporations 9.1 billion annually (Richter, 2019). According to the Business Software Alliance, 39% of software installed on personal computers worldwide in 2015 was not properly licensed (Richter, 2019). Previous studies have used various theories from other disciplines (sociology, psychology, criminology, etc.) to help explain why individuals commit cybercrimes. The neutralization theory, developed by Skyes and Matza (1957), has been used to explain individual origins of cybercrimes and the justifications behind them (Smallridge & Roberts, 2013). Donner (2016) conducted an investigation that found empirical evidence to support this theory in regards to the self-control of men's and women's intentions to commit software piracy.

Importance of Secure Coding

The importance of secure coding has evolved within the last five years. Figure 1 represents 2018 biggest breaches and vulnerabilities ranked by the number of people impacted.

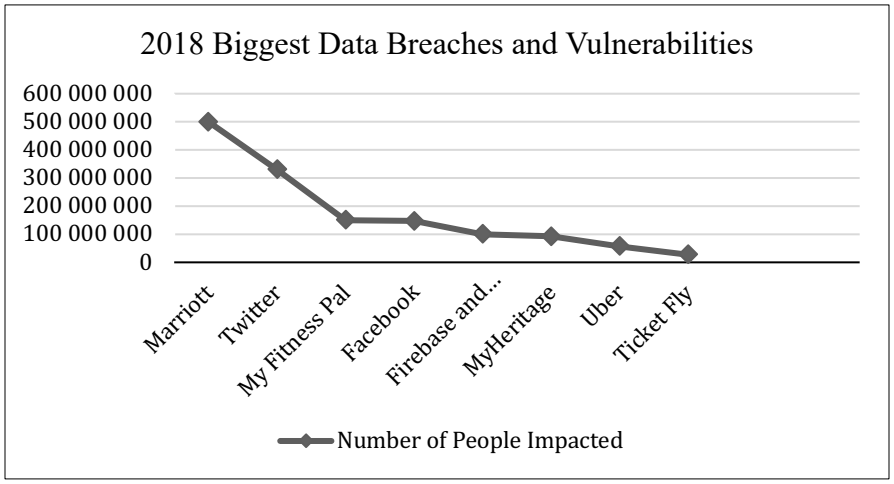

Figure 1. 2018 Biggest Breaches and Vulnerabilities (Abril, 2019).

According to Abril (2019), more than a billion people were affected by corporate data breaches in 2018. These attacks reveal that companies are vulnerable and not immune to data breaches. The hacks on these companies that hold huge amounts of data supports the theory that businesses and organizations are not doing enough to ensure data security.

The Identity Theft Resource Center (ITRC) disclosed that the number of breach incidents in 2018 compared to 2017 is fewer, but the number of records that contained personal identifiable information increased by 126 percent. These results indicate that more secure methods need to be developed to protect usernames and passwords.

RESEARCH METHODOLOGY

The main research question that this study addressed was: Does introducing proactive controls to computing students change their conception of software application security vulnerabilities? The two specific research questions that this study addressed were:

RQ_1: By implementing the OWASP top 10 proactive controls, will users have a new conception towards reducing the possibilities of encountering security vulnerabilities?

RQ_2: By comparing the same data set of students but in two different time settings, will there be a significant change of conceptual view from before being introduced to the OWASP top 10 proactive controls to after they have completed their learning of the OWASP 10 proactive controls?

Research Instruments

All of the constructs were measured with previously validated instruments. In this study, two surveys were conducted on students pursuing an associate's degree in Computer Programming (CP). The instruments used in the study were a series of survey questions that were measured on a 4-point Likert-type scale in which 1 denoted "strongly disagree (SD)," 2 denoted "disagree (D)," 3 denoted "agree (A)," and 4 denoted "strongly agree (SA)." The participants for the surveys were sent a link through email to access the questionnaire between February 2019 and March 2019. The targeted participants were students who were in their 2nd quarter (freshman year). Participants were first given an introduction and the purpose of the survey before being asked to take it. Participants were expected to fully understand the purpose of the survey and agree to the terms and conditions before proceeding to completing the survey. IRB approval was obtained prior to recruitment of subjects.

The purpose of the survey was to collect data and analyze the results to compare the students' perceptions of software application security before and after being introduced to the OWASP top 10 proactive controls. Surveys and questionnaires are widely used in research to target a particular population with designed questions to measure and collect data pertaining to a specific topic (Alvarado, León, & Colón, 2016). This technique provides precise calculations of the variables that are being used in the study.

Questionnaire Development

The questionnaire was divided into three main categories. The first category was the student's opinion on the importance of security risks when developing software applications. The second category was the student's opinion of the OWASP top 10 proactive controls. The third category was the student's understanding of the timeframe of addressing and implementing security features in the life cycle of a software development project.

Questionnaire Testing

A pretest was conducted on a group of participants to complete the questionnaire by themselves, without intervention or support from the researcher. This test was conducted through a survey questionnaire created on www.surveymonkey.com. A link to the questionnaire was sent out to the participants through email with an introduction text explaining the terms and the estimated time of completion. The researchers used SurveyMonkey due to its reputation of stability and for the simple appearance of the interface that it provides. SurveyMonkey uses traditional web widgets such as checkboxes, radio buttons, etc. This interface helped reduce the amount of instructions on how to reply to the questions.

Questionnaire Deployment
The approach to invite participants to the survey was done online through the college's email service. The invitation had an important role as an initial contact with the participants because it explained the purpose of the research, the researchers, the college that was involved, and the average time that would be spent to complete the questionnaire. This was obtained by the average time of the respondents who participated in the pretest. SurveyMonkey.com provided the header of the survey, whether it was the pre-knowledge or post-knowledge survey questionnaire for the Computer Programming students. This helped show the participant that the research was focused on a specific part of the student population at the college.

RESULTS

The first questionnaire was taken by 37 students currently enrolled in the Associate of Science (A.S.) in Computer Programming (CP) in the School of Information Systems and Technology. The students who participated in this questionnaire were in their second quarter term (first six months of enrollment) at Pittsburgh Technical College located in Oakdale, Pennsylvania. The first questionnaire was emailed in February, 2019, and the data was collected until March 1, 2019. The second questionnaire was emailed in March, 2019, and the data was collected until the end of the month. The second questionnaire was taken by 24 of the 37 students who previously took the first questionnaire. These 24 students were educated on the OWASP top 10 proactive controls.

Findings
The following statements in the questionnaire on the pre-knowledge and post-knowledge on the OWASP top 10 proactive controls were given to the participants of this research study:

RQ_1: I consider the possible security risks when developing software applications.
RQ_2: I feel that secure programming is a requirement for programmers to implement.
RQ_3: I feel that secure programming is dependent on the style of the programmer.
RQ_4: I believe that exposure of private information is the highest ranking vulnerability of an application.
RQ_5: I believe that deserialization of untrusted data is the highest ranking vulnerability of an application.
RQ_6: I believe that insufficient logging and monitoring is the highest ranking vulnerability of an application.
RQ_7: I think software attackers have many different paths through my application to cause harm to an organization.
RQ_8: I believe that common security vulnerabilities have changed over the last 5 years.
RQ_9: I believe that security can be implemented at all levels of development.
RQ_{10}: I believe that the requirements of the application should be completed before security concerns are addressed.

Below are two tables that display the results of students' pre-knowledge and post-knowledge of software application security. Table one represents students' pre knowledge and table two represents students' post knowledge.

Table 1. Students' Pre-Knowledge of Software Application Security.

RQ	SA	A	D	SD
RQ_1	6	25	5	1
RQ_2	15	10	8	4
RQ_3	26	7	2	2
RQ_4	14	11	7	5
RQ_5	6	19	12	0
RQ_6	5	15	14	3
RQ_7	16	15	5	0
RQ_8	19	13	5	0
RQ_9	24	9	4	0
RQ_{10}	10	12	8	6

Table 2. Students' Post-Knowledge of Software Application Security.

RQ	SA	A	D	SD
RQ_1	8	14	1	1
RQ_2	18	5	0	1
RQ_3	12	10	1	1
RQ_4	12	12	0	0
RQ_5	5	11	8	0
RQ_6	6	10	6	2
RQ_7	14	9	1	0
RQ_8	11	8	4	1
RQ_9	16	6	1	1
RQ_{10}	12	8	2	2

A noticeable change of students' pre-knowledge and post-knowledge of software application security can be observed through the results in research questions two, four, and ten. The results from the other research questions did not produce a significant amount of difference from the students' pre-knowledge and post-knowledge of software application security. It is important to observe that the majority of the participants selected "strongly agree" or "agree" in considering security risks when they developed software applications (research question one). Another important observation to make is the decrease in the perception from the participants who selected "disagree" from the pre-knowledge to the post-knowledge of software application security.

In research question two, 67% of the participants in the pre-knowledge questionnaire selected either "strongly agree" or "agree" for their perception that secure programming is a requirement for programmers to implement. In addition, 95% of the participants in the post-knowledge questionnaire selected either "strongly agree" or "agree" for their perception that secure programming is a requirement for programmers to implement. This indicates that a difference of 28% of students' perceptions changed between the pre-knowledge and post-knowledge questionnaire.

In research question four, 68% of the participants in the pre-knowledge questionnaire selected either "strongly agree" or "agree" for their perception that that exposure of private information is the highest ranking vulnerability of an application. Of the participants, 100% in the post-knowledge questionnaire selected either "strongly agree" or "agree" for their perception that exposure of private information is the highest ranking vulnerability of an application. This indicates that a difference of 32% of students' perceptions changed between the pre-knowledge and post-knowledge questionnaire.

In research question four, 59% of the participants in the pre-knowledge questionnaire selected either "strongly agree" or "agree" for their perception that the requirements of the application should be completed before security concerns are addressed. In addition, 83% of the participants in the post-knowledge questionnaire selected either "strongly agree" or "agree" for their perception that the requirements of the application should be completed before security concerns are addressed. This indicates that a difference of 24% of students' perceptions changed between the pre-knowledge and post-knowledge questionnaire.

DISCUSSION

The results from the pre-knowledge and post-knowledge questionnaire on the participants' perceptions of their belief that insufficient logging and monitoring is the highest ranking vulnerability of an application did not have a sufficient change. This result could be an implication that the participants did not change their perception after learning the OWASP top 10 proactive controls and software application security vulnerabilities. The results indicated that the majority of the participants selected "strongly agree" or "agree" in considering the many different paths that software attackers can take to cause harm to an organization. Another important observation to make from the results is the decrease in the perception from the participants' who selected "disagree" from the pre-knowledge to the post-knowledge of software application security.

Limitations

The implementation of this study was not without certain limitations. The study was limited by the fact that it only focused on the measurement of students' pre-knowledge and post-knowledge of software application security through one delivery method on the OWASP top 10 proactive controls. The study is limited in not being able to control a variety of large variables, such as learner characteristics, instructional method, teacher involvement, and student interactions. The results of this study were limited to a specific group of participants in a single college. These results should not be considered generalized across different universities and countries.

Another limitation of this study was the size of the data sample. Further investigation is needed to establish if the same results can be duplicated through a larger data sample and applied across a broader context of universities. The participants being all first year students in a computer programming associate degree program may not be a good representative sample of a broader university student population. Similarly, the instructor involved in the delivery method of the OWASP top 10 proactive controls and software application security vulnerabilities had a high degree of experience in cybersecurity that may have influenced these results. Another limitation is the fact that limited insight into the participants' pre-knowledge and post-knowledge on software application security vulnerabilities was obtained.

CONCLUSION

In conclusion, our study gained insightful information about the students' pre-knowledge and post-knowledge of secure coding best practices using OWASP proactive controls. Students' perceptions of software application security vulnerabilities changed after they were educated on the importance of code security. Based on the findings, 65% of students who participated in the survey were not knowledgeable about OWASP top 10 proactive controls. This indicates that educating code security concepts to Computer Programming students should be introduced early in their software development education program. Code security best practice should be taught and demonstrated by setting up simulations that mimic real world scenarios. This would allow students to interface with the realities of software application vulnerabilities and learn the difference between secure coding and unsecure coding. Students should be given ample time to work in development environments during their internships or practicum so that they may be exposed to code security best practices other than theories.

In addition, the study concluded that teaching college students the importance of code security is best implemented when students are educated about the dangers of unsecure software development through examples of trending software applications that have been hacked and showing them statistics of organizations that have been impacted by data breaches. After learning about code security, nearly 100% of students who participated in the survey became aware of OWASP's top 10 proactive controls and learned how to implement code security best practice in their software development. Students were educated about software applications that handle data such as credit card information, personal information, medical records, and bank information. The findings of this study indicated that the participants' pre-knowledge and post-knowledge of application security changed after being educated and introduced to the OWASP top 10 proactive controls of secure coding. Students' post-knowledge of software application security suggested that they were becoming aware of the importance of code security in the software development lifecycle.

REFERENCES

Aljawarneh, S. A., Alawneh, A., & Jaradat, R. (2017). Cloud security engineering: Early stages of SDLC. *Future Generation Computer Systems*, *74*, 385-392.

Anton, K.., Manico, J., & Bird, J. (2018). 10 critical security areas that software developers must be aware of. Retrieved from https://www.owasp.org/images/b/bc/OWASP_Top_10_Proactive_Controls_V3.pdf

Abratt, R., Nel D., & Higgs, S. N. (1992). An examination of the ethical beliefs of managers using selected scenarios in a cross-cultural environment. *Journal of Business Ethics*, *11*(1), 29-35.

Benham, H. C., & Wagner, J. L. (1995). Ethical attitudes of business students and MIS personnel. *Proceedings of the ACM SIGCPR Conference* [online], Nashville TN USA, 44-49. Retrieved from www.acm.org/pubs/articles/proceedings/cpr/212490/p44-wagner/p44-wagner.pdf

Bishop, M. (2010). A clinic for "secure" programming. *IEEE Security Privacy, 8*(2), 54-56.

Chahar, C., Chauhan, V. S., & Das, M. L. (2012). Code analysis for software and system security using open source tools. *Information Security Journal: A Global Perspective*, *21*(6), 346-352.

Alvarado, F. C., León, M. P., & Colón, A. O. (2016). Validation of a questionnaire on research-based learning with engineering students. *JOTSE*, *6*(3), 219-233.

Culp, S. (2019) Cybercrime: A major threat to trust in the digital economy. Retrieved from https://www.forbes.com/sites/steveculp/2019/03/25/cybercrime-a-major-threat-to-trust-in-the-digital-economy/#7e5dd7232cb7

Danielle, A. (2019). These were the worst data breaches and vulnerabilities of 2018. Retrieved from http://fortune.com/2019/01/04/worst-data-breaches-of-2018/

Donner, C. M. (2016). The gender gap and cybercrime: An examination of college students' online offending. *Victims & Offenders*, *11*(4), 556-577.

Gates, B. (1995). *The road ahead*. New York, NY: Viking Penguin Group.

Givon, M., Mahajan, V., & Muller, E. (1995). Software piracy: Estimation of lost sales and the impact on software diffusion. *Journal of Marketing, 59*(1), 29- 37.

Howard, M. (2004). Building more secure software with improved development processes. *IEEE Security & Privacy*, *2*(6), 63-65.

Kissel, R. L., Stine, K. M., Scholl, M. A., Rossman, H., Fahlsing, J., & Gulick, J. (2008). *Security considerations in the system development life cycle* (No. Special Publication (NIST SP)-800-64 Rev 2).

McKelvey, N. (2012). Developing a secure programming module to cope with modern vulnerabilities. *International Journal of Information and Network Security, 1*(1), 41.

Richter, Felix. (2019). The cost of software piracy. Retrieved from https://www.manufacturing.net/data-focus/2016/07/cost-software-piracy

Smallridge, J. L., & Roberts, J. R. (2013). Crime specific neutralizations: An empirical examination of four types of digital piracy. *International Journal of Cyber Criminology*, *7*(2).

Seacord, R. C., & Rafail, J. A. (2006). Secure coding standards. *Proceedings of the Static Analysis Summit, NIST Special Publication*, *13*, 17.

Sykes, G. M., & Matza, D. (1957). Techniques of neutralization: A theory of delinquency. *American Sociological Review, 22*(6), 664-670.

Von Solms, R., & Van Niekerk, J. (2013). From information security to cyber security. *Computers & security*, *38*, 97-102.

Warren, T., Favole, J., Haber, S., & Hamilton, E. (2016). Cybercrime costs more than you think. *Hamilton Place Strategies Report*.

Whitney, M. P. (2015). *ESIDE: The integration of secure programming education into the IDE* (Doctoral dissertation, The University of North Carolina at Charlotte).

Whitney, M., Lipford, H. R., Chu, B., & Thomas, T. (2018). Embedding secure coding instruction into the IDE: Complementing early and intermediate CS courses with ESIDE. *Journal of Educational Computing Research*, *56*(3), 415-438.

Zhu, J., Xie, J., Lipford, H. R., & Chu, B. (2014). Supporting secure programming in web applications through interactive static analysis. *Journal of Advanced Research*, *5*(4), 449-462.