Reflective Assignment

Without a doubt, the Secure Software Development has been my most challenging – and yet, rewarding – module in this program. Over the course of my studies, I have found that I reflect best by applying Rolfe's "What"/So What?/Now What?" model and incorporating some of Moon's analytical techniques (Fallin, 2020; Moon, 2004; Rolfe et al., 2001).

What?

This module began by reintroducing the Unified Modelling Language (UML) as the precursor to well formed – and thus, inherently more secure – software (Peoples, 2022a). The theme that *well designed, well built,* and *well tested* applications suffer from fewer vulnerabilities carried through the entire course and was emphasised strongly in our core textbook (Pillai, 2017), and also evident in Saltzer's and Schroeder's commandments for secure systems (1975). Dr. Peoples built upon this foundation by equipping our class with tools to help write high quality code through the use of linters, testing modules, and testing frameworks (ISO, 2020; OWASP, 2020; Peoples, 2022b). This course taught me about the aspects of programming languages and software that impact security, such as kernel operations, potential for buffer overflows, and "evil" regexes that can rapidly degrade or crash systems (Peoples, 2022c; Weidman, n.d.). We learned about different system architectures and their evolution from generally monolithic constructs to distributed arrangements of microkernels and microprocesses that leverage advances in computing and network performance (Bucchiarone et al, 2018; Fritzsch et al., 2019; Pillai, 2017).

Throughout this module, I worked with a team on two primary assignments: a design document for a secure web application, and subsequently, the resultant application and documentation. This culminated in a demonstration of our secure web application and evaluation of our submitted application artifacts.

So What?

In summary, this module led me to conclude that application security cannot be ascertained through a single solution; rather, application security exists when the conditions from all interrelated components minimise vulnerabilities. Human behaviour, technological limitations, and trade-offs between security and useability mean that no system will ever be fully secure.

Learning that high quality software is easier to secure is a concept whose simplicity betrays its importance. One of my biggest take-aways from this course is that better code directly correlates with more secure software. This course – through Codio, and especially through the Coding Output summative assignment – gave me the opportunity to expand my applied programming experience. As a hands-on learner who truly enjoys working

with teams, the group assignments allowed me to apply the lessons learned throughout the module, and to build upon that knowledge by pursuing independent research. Our team iterated between designing software, implementing the design in code, and revisiting our design when we hit a roadblock or found an efficiency. This “trial by combat” was a positively enlightening experience, and I learned just how useful collaborative environments, Kanban, and Agile scrum project management techniques are when working in geographically separated teams.

Another significant lesson I learned from this course is that building a secure application is not easy, nor is it a fait accompli after initial development. Software is inherently dependent on continuously evolving components such as open-source libraries, operating systems, networks, firmware, and hardware. Developers must remain vigilant of emerging vulnerabilities by continuing to test and update software throughout its lifecycle to remain secure.

Now What?

Application security is not for the faint of heart; it requires a thorough understanding of multiple technical domains. This course has given me a roadmap to becoming an effective practitioner of secure software development through the following activities:

Continue learning to write better code

As with any skill, expertise comes with time and effort. I plan to improve this skill in two ways: 1) reading through good and bad examples of open-source applications to understand their implementation of security techniques, and 2) apply the skills learned to begin my own secure development projects. Just like learning human languages, continued exposure and immersion with programming helps me improve my ability to understand the code written on the screen *as well as* the subtext of that code (e.g., identifying “code smells,” dangerous expressions, and programming inefficiencies). Linters have also proven to be useful training aids. Using tools such as Pylint to check modules of code throughout development has already helped me mitigate vulnerabilities before testing and has simultaneously helped me write efficient, easier-to-read code.

Expand knowledge of current and emerging vulnerabilities

Formally studying computer science has revealed a thriving community dedicated to the advancement of the field. Organisations such as the Open Web Application Security Project (OWASP, 2022a), the Institute for Security and Open Methodologies (ISECOM, 2022), the CVE Program (CVE Program, 2022), and the National Institute of Standards and Technology (NIST, 2022a) provide excellent resources on security tactics, techniques, and procedures, and share vulnerability information with the community. There are also many useful podcasts, newsletters, and YouTube channels that discuss the evolving cyber threat landscape. Investing time to learn about the origins and evolution of exploits and vulnerabilities will make it easier to stay abreast of new threats.

Practise penetration testing

Analysing software vulnerabilities from an adversarial perspective will enable me to anticipate potential attack vectors during the development phase. As I gain more experience identifying vulnerabilities in applications through automated and manual detection techniques, I will learn to identify patterns and indicators of potential weaknesses that may expose data, allow for cross-site scripting, reveal plain-text login credentials, or any other software behaviour that will compromise its intended performance. Furthermore, because I attended the Network Security (NS_PCOM7E) simultaneous to this course, I quickly realised how resources such as the OWASP Top Ten (OWASP, 2022b), Common Vulnerabilities and Exposures database (CVE Program, 2022), and National Vulnerability Database (NIST, 2022b) could be used for both offensive and defensive purposes. To safely practise penetration testing, I have established an account with an online cyber target range called Hack The Box that allows me to connect to their network to reconnoitre and exploit vulnerable systems (Hack The Box, n.d.). I have also created my own local test range by using virtual machine images designed with vulnerabilities from VulnHub (2022).

References

Bucchiarone, A., Dragoni, N., Dustdar, S., Larsen, S.T. and Mazzara, M. (2018). From Monolithic to Microservices: An Experience Report from the Banking Domain. *IEEE Software*, 35(3), pp.50–55. doi:10.1109/ms.2018.2141026.

CVE Program (2022). *CVE® Program Mission.* [online] Available at: https://www.cve.org. [Accessed 04 September 2022].

Fallin, L. (2020). *LibGuides: Reflective writing: Rolfe*. [online] libguides.hull.ac.uk. Available at: https://libguides.hull.ac.uk/reflectivewriting/rolfe. [Accessed 03 September 2022].

Fritzsch, J., Bogner, J., Zimmermann, A. and Wagner, S. (2019). From Monolith to Microservices: A Classification of Refactoring Approaches. *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, [online] pp.128–141. doi:10.1007/978-3-030-06019-0_10.

Hack The Box. (n.d.). *Hacking Training For The Best*. [online] Available at: https://www.hackthebox.com/. [Accessed 04 September 2022].

ISECOM. (2019). *ISECOM*. [online] Available at: https://www.isecom.org/. [Accessed 04 September 2022].

ISO. (2022). *ISO/IEC/IEEE 29119-4:2015 Software and systems engineering — Software testing — Part 4: Test techniques*. [online] Available at: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:29119:-4:ed-1:v1:en [Accessed 04 September 2022].

Moon, J. A. (2004) *A Handbook of Reflective and Experiential Learning : Theory and Practice*. London: Routledge. Available at: https://search-ebscohost-com.uniessexlib.idm.oclc.org/login.aspx?direct=true&db=nlebk&AN=102749&site=ehost-live. [Accessed 03 September 2022].

NIST (2022a). *National Institute of Standards and Technology | NIST*. [online] NIST. Available at: https://www.nist.gov/. [Accessed 04 September 2022].

NIST (2022b). *National Vulnerability Database*. [online] NIST. Available at: https://www.nist.gov/. [Accessed 04 September 2022].

OWASP. (2020). *OWASP Web Security Testing Guide*. [online] Available at: https://owasp.org/www-project-web-security-testing-guide/. [Accessed 03 September 2022].

OWASP (2022a). *OWASP Foundation, the Open Source Foundation for Application Security.* [online] owasp.org. Available at: https://owasp.org/. [Accessed 03 September 2022].

OWASP (2022b). *OWASP Top Ten.* [online] owasp.org. Available at: https://owasp.org/www-project-top-ten/. [Accessed 04 September 2022].

Peoples, C. (2022a) *Unit 1: Introduction to Secure Software Development.* [online] University of Essex Online. Available at: https://www.my-course.co.uk/course/view.php?id=8478&section=8. [Accessed 03 September 2022].

Peoples, C. (2022b) *Unit 5 Lecturecast: Testing.* [online] University of Essex Online. Available at: https://www.my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%203/content/index.html#/. [Accessed 03 September 2022].

Peoples, C. (2022c) *Unit 7 Lecturecast: Operating System Security.* [online] University of Essex Online. Available at: https://www.my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%204/content/index.html#/. [Accessed 03 September 2022].

Pillai, A. B. (2017). *Software Architecture with Python*. Birmingham, England: Packt Publishing.

Rolfe, G., Freshwater, D., Jasper, M. (2001) *Critical reflection in nursing and the helping professions: a user's guide.* Basingstoke: Palgrave Macmillan.

Saltzer, J.H. and Schroeder, M.D. (1975). The protection of information in computer systems. *Proceedings of the IEEE*, 63(9), pp.1278–1308. doi:10.1109/proc.1975.9939.

VulnHub. (2022). *Vulnerable By Design ~ VulnHub*. [online] Available at: https://www.vulnhub.com/. [Accessed 04 September 2022].

Weidman, A. (n.d.). *Regular expression Denial of Service - ReDoS | OWASP*. [online] Available at: https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS. [Accessed 04 September 2022].