UNIT 13: Software Vulnerabilities

Estimated Time in Hours: 10

Big Idea(s)	Enduring Understandings	Projects & Major Assignments
1 Ethics 2 Establishing Trust 5 System Security	5.3	- Research, summarize, and list examples of specific vulnerability types. - Research insecure cryptographic algorithms and provide secure replacements for them. - Create summary videos of how security best practices mitigate or prevent software vulnerabilities. - Advanced students will code and demonstrate buffer overflow vulnerabilities in the C programming language.

Guiding Questions:

- Why are CVE and OWASP beneficial to the security community?
- What is an injection attack? What does it do?
- How can buffer overflow attacks be prevented?
- Is cryptography, regardless of what kind, a guaranteed way to secure data?
- What is the SSDLC and how does it differ from old software development methods?
- What is the difference between static and dynamic analysis?
- What are zero-day attacks and why are they so devastating?
- Are zero-day attacks always discovered by adversaries?
- Why is patching so important?
- How is process isolation essential to security?

Learning Objectives & Respective Essential Knowledge Statements	Materials	Instructional Activities and Classroom Assessments
5.3 EU: Security vulnerabilities in software are weaknesses in a system's design, implementation, or operation and management	• Computer, lecture slides, projector, graphic organizers, access to Internet	• Review software vulnerabilities. • Ask students why software is vulnerable (review of Unit 12).

that could be exploited to violate the system's security policy.		
5.3.1 LO: Students will describe common security-related software vulnerabilities. 5.3.3b EK: Security vulnerability reports such as Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE) are publicly available for software systems and should be monitored, or subscribe to their alerts.	• CVE Explanation in 90 seconds: "What is Common Vulnerabilities & Exposures (CVE)." *YouTube*, uploaded by F5 Inc., 2 Feb 2020, https://youtu.be/qfpnJyTl1To • CVE Charts: "Vulnerabilities By Type." CVE Details, *CVEDetails.com*, https://www.cvedetails.com/vulnerabilities-by-types.php • OWASP Top 10: "OWASP Top Ten Web Application Security Risks." Open Web Application Security Project, *OWASP.org*, https://owasp.org/www-project-top-ten/	• Introduce the Common Vulnerabilities and Exposures (CVE) as a way to categorize, track, and share information about vulnerabilities. • Show the linked YouTube video explaining CVEs. Use a video viewing guide. • Show the CVE statistics. Ask students why they think certain vulnerability types are more prevalent and how those change over the years. • Explain the OWASP (Open Web Application Security Project) Top 10 security risks against web applications. • Challenge students with an independent activity to research, summarize, and list examples of specific vulnerability types.

5.3.1a EK: Injection attacks occur when an external source such as a user provides input that causes a program to behave in ways that violate the security policy by executing harmful commands. 5.3.1c EK: A software vulnerability may exist when data is allowed to include unauthorized control instructions that dictate how the program should behave and thus can cause the program to behave in a way that violates the security policy.	• OWASP #1 Vulnerability (Injection Attacks): "OWASP Top 10: Injection Attacks." *YouTube*, uploaded by F5 DevCentral, 13 Dec 2017, https://youtu.be/rWHvp7rUka8	• Introduce injection attacks. As many injection attacks focus on SQL injection, you should consider covering SQL database basics as well. • Show the linked YouTube video explaining the OWASP #1 (Injection Attack). Use a video viewing guide to assess their learning. • Ask students how they think injection attacks can be prevented. They will learn more on this later.
5.3.1b EK: A buffer overflow is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations, and how this can be used as an entry point by an attacker to violate security policy.	• Interactives for Buffer Overflow / SQL Injection: "Cybersecurity Interactives." E-Mate 2.0, *e-mate-bbc.org*, https://s3.amazonaws.com/e-mate2/Cybersecurity+Interactives/Cybersecurity+Interactives.html • Buffer Overflow overview: "Ethical Hacking: Buffer Overflow Basics."	• Show the linked YouTube video explaining Buffer Overflow attacks. Use a video viewing guide to assess student learning. • Ask students how they think buffer overflows can be prevented or at least reduced. The video covers some tactics.

	YouTube, uploaded by National Consortium for Mission Critical Operations, 24 Nov 2014, https://youtu.be/SOoJcrR4Ijo	
5.3.1d EK: A software vulnerability may exist when cryptographic functions are not implemented properly or when the cryptographic functions are assumed to provide more security than the algorithm provides.		• Review how cryptography can protect systems. Although this may seem like a security catch-all, that can be deceiving. • Emphasize that just because cryptography exists, doesn't mean the data is secure. Cover insecure or improperly setup cryptography. • Task students with researching cryptographic algorithms no longer considered secure. They can summarize why it is no longer considered secure and list any secure replacements.
5.3.1e EK: Changes to the environment can cause software to no longer meet the security policy and secure software must include considerations for how to implement future changes (e.g., credentials, algorithms, and patching code to correct bugs and errors).		• Explain how secure environments require any new software or updates to be verified and tested for both security and functionality. • Ask students whether they think this is good or bad. The process may slow down new additions to the environment, but ultimately it acts to its benefit.
5.3.1f EK: A software vulnerability can occur when external components that don't		• A secure environment will often limit what can be connected to them. For example, they may not be able to

meet the security policy requirements are connected to the system.		connect with USB drives, CDs, or other computer networks. • Ask students why these restrictions are in place. Does it make the environment more secure? How do they enforce it?
5.3.2 LO: Students will identify the processes of developing secure software.	• Secure Software Development Life Cycle: "Secure Software Development Lifecycle." Digital Maelstrom, *DigitalMaelstrom.net*, https://www.digitalmaelstrom.net/security/secure-software-development-lifecycle-ssdlc/	• Introduce a framework for developing secure software, such as the Secure Software Development Life Cycle (SSDLC). • The SSDLC differs from many other frameworks in that it requires security throughout the entire process. Each stage of the original SDLC (without security) now requires essential security actions. • Ask students why it is important to consider security throughout the entire development and operation of software. Relate this to the scenarios in Unit 12.
5.3.2a EK: Input validation is code added to the program that verifies input provided by an external source is the type of input expected and will be processed correctly.	• Student-made video explaining input validation: "Input Validation." *YouTube*, uploaded by CLARK Cybersecurity Curriculum Digital Library, 10 Sep 2015, https://youtu.be/-8bDdrZhj_k	• Review how software developers have to anticipate how end users will abuse their software and use it in other ways. • Task students with creating their own summary videos of how security best practices (i.e., input validation) mitigate or prevent software vulnerabilities. A linked YouTube video is provided to the left as an example.

5.3.2b: EK Static analysis of software is a process in which external tools analyze the code and automatically identify potential security vulnerabilities such as potential buffer overflows.	• Static Analysis explanation: "What Are Static Analysis Tools?" *YouTube*, uploaded by goobar, 19 Oct 2018, https://youtu.be/d_BCGvXbpKs	• Explain static analysis as an important piece of SSDLC. Code should be routinely examined as it is developed. • For more explanation, show the linked YouTube video to overview static analysis. Use a video viewing guide to review. • Showcase some insecure coding practices or functions (e.g., buffer overflows in the C programming language) as examples which static analysis can detect.
5.3.2c EK: Development tools and Integrated software Development Environments (IDE)s provide static analysis tools to check for some types of insecure code such as identifying potential buffer overflows.		• Provide examples of static analysis tools. Ask students to compare and contrast.
5.3.3 LO: Students will describe the process of validating that software remains secure through its lifecycle. 5.3.3a EK: A security analysis is a process that is used to verify a program meets a specified list of security requirements.		• Describe how software needs to be validated and secured even after its development. Ask students why this is important. • The SSDLC is one framework that provides security analysis. Use this or a similar framework as an example.
5.3.3c EK: A zero-day vulnerability is a software security flaw that is unknown to people who should be	• Zero-Day Attack: "Anatomy of an Attack – Zero Day Exploit." *YouTube*, uploaded by	• Even if software is secure at launch, it can become subject to a zero-day vulnerability at any point. This is one reason why constant security is necessary even after software is developed.

responsible for patching or fixing the flaw.	FireEye, Inc., 19 May 2015, https://youtu.be/-BIANfzF43k	• Show the linked YouTube video about zero-day exploits. Why are zero-day attacks so devastating? Are zero-day exploits always found by an adversary?
1.3.3d EK: Disclosure of software vulnerabilities to a party other than the software developer is legal and can be harmful.		• Explain how security researchers search for zero-day vulnerabilities before adversaries. They seek to find and report these bugs before they are reported. • Discuss bug bounties and how companies will often pay for the discovery and discrete disclosure of bugs.
5.3.3d EK: Managing vulnerability reports, patching and patch distribution is a key part of software security.		• Review the importance of updates from Unit 12. Patching is synonymous with updates. Ask students why patching is essential, especially in the wake of zero-day attacks?
5.3.3e EK: Dynamic analysis is a process in which external tools analyze the execution of code in order to automatically identify potential security vulnerabilities.		• Contrast dynamic analysis with static analysis. • Ask students why it is important to test the software in both its execution and raw code states.
2.3.2 LO: Students will know that the principle of process isolation prevents tampering or interference from/by other processes. 2.3.2a EK: A process is a program running on a computer.		• Review processes and the principle of process isolation. • Why is process isolation important to the security of programs?

2.3.2b EK: Each process has a region of the memory (address space), which only it can access. 2.3.2c EK: Processes have to use defined communications mediated by the operating system to communicate with other processes.	• Buffer Overflow programming practice: "CSC 5991 Cyber Security Practice | Lab 2: Buffer Overflows." Wayne State University College of Engineering, *Wayne.edu*, http://webpages.eng.wayne.edu/~fy8421/16sp-csc5991/labs/lab2-instruction.pdf	• Explain how the operating system will distribute memory to the processes. How can this be exploited in attacks like the buffer overflow? • Advanced programming students may practice a buffer overflow example in the C programming language (linked to the left).
8.1.1h EK: Cybersecurity events have led to the development of various cybersecurity career paths and various needs in order to prepare people for these new types of jobs.		• Explore a relevant career, such as secure software assessor.