Hack Your Own Code: Advanced Training for Developers

Abstract

This class provides developers an exciting chance to hone their programming skills while also learning to exploit common web vulnerabilities. Unlike most training, this will not use static demos based on pre-canned source code. Students will program small parts of a larger application during the class's lab periods. After the component has been written, students will review the code for the vulnerability being focused on in the lab. Vulnerable code will be run on a class-accessible server while the instructor guides students through exploiting the vulnerabilities. After the vulnerability has exploited, students will be shown how their own code can be fixed (if it was vulnerable) and the best way to prevent the flaw in the first place.

This full process will be performed for all major code vulnerabilities in the OWASP Top Ten. Exploitation and patching labs (but not programming) will be held for other vulnerabilities, including logic flaws that are hard to represent on the Top Ten. Several labs will feature prizes for the students that first find or exploit the targeted vulnerability. Environments and examples will be setup for all major platforms requested by pre-registered students. Students should bring a laptop with them, preferably with VMWare Player already installed. A virtual machine based on the OWASP Live Boot CD will be provided for lab work. The virtual machine will include development tools, but students should feel free to bring their favorite programs too.

Unlike many classes, this will allow programmers to focus on their own code. This makes the class far more interactive than a typical secure development class. The focus on lab work engage the students and make it a far more memorable experience.

Course Outline

1) Introductions

2) Agenda

3) Secure Development Lifecycle (SDLC)

A cradle to grave development life cycle is outlined. Threat modeling is introduced as a means of ensuring that software design can meet policy needs.

4) Lab: Threat Modeling

Different application scenarios will be described and the class will verbally work through modeling threats to them. The primary goal is to get developers to think like an attacker, allowing them to anticipate threats to their own applications.

5) Principles of Secure Code

Focusing on code quality concepts, the practices that help to quantify secure coding are explored. Practical goals and approaches are reviewed, so that a consistent understanding of "secure" can be encouraged and measured appropriately.

6) Authentication & Authorization

The different aspects of authentication and authorization are covered. Pitfalls and common attacks against identity management are explored. Mistakes covered include insecure direct object references, failure to restrict URL access, and various types of other authentication and authorization bypass.

7) Lab: Access Control

Students will learn to attack several web pages that contain a variety of access control vulnerabilities, including Insecure Direct Object Access and Failure to Restrict URL Access. After the vulnerabilities have been exploited, the source code will be reviewed by the students to pinpoint where the flaw was introduced. Strategies will be discussed for preventing this type of flaw.

8) Mini-Lab: Weak Session Identifiers

A variety of weak and some strong session identifiers will be provided to students, along with tools for gauging their strength.

9) Session Management

Due to the stateless nature of the web, the security implications of session generation and management are discussed. This includes both client-side token tracking and server-side session handling.

10) Lab: SQL Injection

Students will finish a simple web page that generates a report based on user-supplied input. Student code (and vulnerable code provided by the instructor) will be deployed on the class web server and tested for SQL Injection vulnerabilities. Students will be shown how SQL Injection can be exploited to extract data and execute arbitrary system commands.

After the attacks are complete, vulnerable source code will be reviewed as a group to spot where the vulnerability was introduced.

11) Lab: Cross-Site Scripting

Students will finish two simple web pages that store user-supplied input for comments on a blog. Student code (and vulnerable code provided by the instructor) will be deployed on the class web server and tested for Cross-Site Scripting vulnerabilities. Students will be shown how Cross-Site Scripting can be exploited to control a victim user’s browser and supply arbitrary content on the website.

After the attacks are complete, vulnerable source code will be reviewed as a group to spot where the vulnerability was introduced.

12) Input Validation

The heart of securing software is dealing with user-controlled data to ensure that it doesn't violate the integrity of a computer system. Improper input validation can allow for vulnerabilities like Cross-Site Scripting and SQL Injection, which are covered extensively. Where relevant, buffer overflow attacks will be covered. Less common input validation vulnerabilities such as XML Injection, XML Entity Expansion, XPATH Injection, and LDAP Injection are also discussed. The advantages of white-listing over blacklisting are explained, and examples are provided of when more flexible validation schemes are required.

13) Lab: SQL Injection Patching

The source code from the SQL Injection lab will be revisited and the vulnerabilities will be patched and tested.

14) Lab: Cross-Site Scripting Patching

The source code from the Cross-Site Scripting lab will be revisited and the vulnerabilities will be patched and tested.

15) Proper Encryption

Initialization vectors, key generation and storage, cipher selection, and decryption oracles will all be discussed. Hashing and secure password storage will also be explained.

16) Mini-Lab: Hash Breaking

A set of insecurely generated password hashes will be provided to the students along with tools used for password attacks. Students will be shown how easy and fast it is to obtain plaintext passwords from insecure storage.

17) Logic Flaws

Application logic flaws can be devastating, but may take no special technical skills to exploit. Preventing them during the design and implementation phases will be discussed, as will techniques for finding logic flaws in existing applications.

18) Lab: Logic Flaw Exploitation

Students will be given access to several webpages with logic flaws on the class server.

19) Other Attacks

This module explores additional vectors of attack such as Cross-Site Request Forgery, insecure redirects, HTTP response splitting, browser specific issues, and rich media security. Compound and other advanced attacks are also covered in this module.

20) Mini-Lab: XML Attacks

Students will be given the opportunity to interact with several webpages that accept XML input. A number of XML attacks will be possible, including XML Injection, XML Bombs, and XML System Entity Expansion.

21) Security Hygiene

Handling exceptional circumstances poorly can leak information about a system useful to an attacker, and in some cases be a source of compromise themselves. This module outlines a variety of concerns and best practices in the logging and communication of errors.

22) Final Lab: Hacking Contest

All remaining time will be used for students to test their skills against an intentionally vulnerable web application. The student that discovers the most vulnerabilities will receive the grand prize!

Instructor Bio

Mike Park is a Managing Consultant at Trustwave. He is a member of Trustwave's SpiderLabs - the advanced security team focused on penetration testing, incident response, and application security. He has over 12 years experience building and securing software for a variety of companies. Mike is a CISSP and specializes in application security assessment, penetration testing, reverse engineering and secure development life cycle. Mike is an active member of the Ottawa ISSA.