

CompTIA.

CompTIA PenTest+

Exam PT0-002

CompTIA PenTest+ Exam PT0-002

Lesson 13



Web Application-Based Attacks

Objectives

- Given a scenario, perform active reconnaissance.
- Given a scenario, research attack vectors and perform application-based attacks.
- Given a scenario, perform a social engineering or physical attack.
- Explain common attacks and vulnerabilities against specialized systems.
- Given a scenario, perform post-exploitation techniques.
- Explain use cases of the following tools during the phases of a penetration test.

Lesson 13

Topic 13A

Recognize Web Vulnerabilities

Outlining the OWASP Top 10

- Web applications (apps) interact with many different users at the same time, which gives attackers an easy target
- In general, vulnerabilities include:
 - Poorly implemented or non-existent security configurations.
 - Failings in authentication and authorization components.
 - The potential for various types of code injection attacks
- Most common vulnerabilities are described in the OWASP Top Ten

Exposing Sensitive Data

- Security measurements for web resources may leave gaps that leave data exposed and vulnerable.
- Vulnerabilities include insecure default configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages
- This exposure could provide the attacker with further details to research and other vectors to analyze and possibly exploit
 - This is referenced as OWASP Top Ten A6:2017-Security Misconfiguration.

Improperly Handling Errors

- Developers add code to handle errors, ensure continuity, and provide informative error output.
- However, error output can provide the team with key details about the underlying technology.
- Other times the error contains references to the location of files related to the web application
 - This is referenced as OWASP Top Ten A3:2017-Sensitive Data Exposure.

Missing Input Validation

- When user-supplied data is processed by the web application without proper validation, this can leave the system exposed
- Missing input validation can lead to injection attacks, such as SQL, NoSQL, OS, and LDAP injection
- This can then allow a malicious actor to execute commands or access data without authorization
 - This is referenced as OWASP Top Ten A1:2017-Injection.

Signing and Verifying Code

- In order to confirm that a script or executable has not been tampered with, developers create digital signature
 - Using the developer's private key
- If the code is not properly signed, this could allow someone to modify the code, which can result in the following:
 - A malicious actor can gain access to a protected resource.
 - Users may the inadvertently run the invalid or inauthentic code.

Causing a Race Condition

- Race conditions occur when the outcome from execution processes is *directly dependent* on the order and timing of certain events.
- Issues can arise if these events fail to execute in the order and timing intended by the developer.
- For example, an app can check that a file exists and then use it later.
 - You may be able to replace the file *after* it is checked but hasn't been used.
 - This can then trigger app instability or privilege escalation.

Review Activity: Recognize Web Vulnerabilities

- Outline the significance of the OWASP Top 10
- List ways developers can inadvertently expose sensitive data
- Review the consequences of improperly handling errors
- Discuss what could happen when there is a lack of input validation
- Explain why it's good practice to sign and verify code, scripts and executables
- Describe what happens in a race condition

Lesson 13

Topic 13B

Launch Session Attacks

Hijacking Session Credentials

- A **cookie** is a text file that contains the session ID (SID) for that web
 - Used as an authentication token instead of having to re-enter credentials
- **Session hijacking** is stealing the session credential from a user's browser and then use it to *impersonate* the user on a website.
- **Session fixation** requires the user to authenticate with a *known session identifier* that will then be used for impersonation.
- **Session replay** is when the malicious actor gains access to a victim's credentials and uses them to gain access to a session.

Crafting Request Forgery Attacks (XSRF/CSRF)

- A XSRF/CSRF attack takes advantage the trust that remote sites have in a user's system to execute commands on the user's behalf.
 - Takes advantage of the saved authentication data stored inside the cookie to gain access to a web browser's sensitive data.
- For example, the target page has a form with a *Remember Me* check box, which then creates a stored cookie
 - The cookie is then used for authentication when they access the site.
- You can exploit this trust and leverage the user's privileges

Escalating Privilege

- **Horizontal Privilege Escalation** - gaining access to an account with *different access or permissions* than the one currently in use.
 - This approach reduces the risk of raising suspicion
- **Vertical Privilege Escalation** - gaining access to an account with *higher* privileges than the one currently in use
 - Use the account to gain access to normally protected resources
 - For instance, a process might run with local administrator privileges, but a vulnerability allows the arbitrary code to run with higher system privileges.

Upgrading a Non-Interactive Shell

- A non-interactive (or restricted) shell is limited in use, in that, unlike an interactive shell, has minimal functionality:
 - You can't press the up-arrow key to display prior commands; or when pressing TAB, the command does not auto-complete
 - You can't move through the system, discover directories, or redirect output.
- When PenTesting the team may need to upgrade a non-interactive shell to an interactive one using a command or script

Exploiting Business Logic Flaws

- Business logic flaws are vulnerabilities that arise from faulty design and implementation issues that can lead to unintended behavior.
 - For example, a poorly implemented method to lock accounts after successive failures to authenticate.
 - Common types of services that are exploited due to business logic flaws are APIs, that include RESTful, XML-RPC and SOAP
- Business logic flaws should be identified and mitigated as they can result in unauthorized access to protected content.

Review Activity: Launch Session Attacks

- Compare the different forms of session hijacking
- Provide an overview of Request Forgery Attacks
- Discuss ways the team can escalate privilege
- Outline why the team would need to upgrade a non-interactive shell
- Explain why it's essential to identify and mitigate business logic flaws

Lesson 13

Topic 13C

Plan Injection Attacks

Identifying SQLi Vulnerabilities

- An SQLi attack is one of the most common types of code injection. As a result, the team should test for this vulnerability
 - One simple method is to use the **single quote method** then look for errors.
- Certain web app APIs allow you to stack multiple queries in the same call.
 - You can use this to obtain data from other tables that might not be directly exposed by the app.
- **Blind SQL injection** is injecting SQL when the web application's response does not contain the result of the query.

Traversing Files Using Invalid Input

- Directory traversal is accessing a file from a location that the user is not authorized to access.
 - The simplest example involves sending a `..\` or `../` command to the app or API.
 - It's most effective when you're able to navigate all the way back to the root and execute *any* command or program in *any* folder on the computer.
- Properly configured web servers will filter out known untrusted input like the directory traversal character set.
 - However, you may be able to bypass these filters by encoding characters in your requests in hexadecimal.

Injecting Code

- Injection attacks can compromise an app in several ways:
 - Cause a DoS, escalate privileges, deface a website
 - Install malicious software on the server hosting the app
- **Code injection** introduces malicious code into a vulnerable application to compromise the security of that application.
- **Command injection** supplies malicious input to the web server, which then passes this input to a system shell for execution.

Executing a Cross-Site Scripting (XSS) Attack

- A XSS attack is an attack which injects JavaScript that executes on the client's browser, and can result in the following:
 - Change the contents of a page, steal session cookies, read sensitive information or inject malware that can execute on the user's computer.
- The three categories of XSS attacks are as follows:
 - **Persistent** - injected code is permanently stored on the web server
 - **Reflected** - code reflects from victim to server then back to attacker
 - **DOM-based** - takes advantage of a web app's client-side implementation of JavaScript to execute the attack solely on the client.

Providing Responses via a Web Proxy


- A web proxy is an intermediary between the client and server
 - The proxy receives web requests and forwards the request to the server
 - The client never has direct contact with the server, which improves security
- Advanced proxy servers today can include firewall and web filters, along with web caching and helps protect against attacks
- As a result, if the customer is using a web proxy, the team need to adjust the PenTest accordingly.

Review Activity: Plan Injection Attacks

- Outline why the team should test for SQLi vulnerabilities
- Review how to traverse files using invalid input
- Discuss how injection attacks can compromise an app
- Describe some of the types of XSS attacks
- Explain why the team may need to adjust the PenTest if the customer is using a web proxy

Lab Activity

Assisted Lab: Using SQL Injection

- Lab types
 - Assisted labs guide you step-by-step through tasks
 - Applied labs set goals with limited guidance
- Complete lab
 - Submit all items for grading and check each progress box
 - Select “Grade Lab” from final page
- Save lab 
 - Select the hamburger menu and select “Save”
 - Save up to two labs in progress for up to 7 days
- Cancel lab without grading
 - Select the hamburger menu and select “End”

Lesson 13

Topic 13D

Identify Tools

Overview of Web Testing Tools

- Testing tools range in capabilities:
 - Simple vulnerability scanners for a particular web application
 - Credential lookup tools and proxies that allow you to manipulate and fine-tune requests
- Examples include:
 - **WPScan** - Automatically gathers data about a WordPress site and compares findings such as plugins against a database of known vulnerabilities.
 - **CrackMapExec** - Post-exploitation tool to identify vulnerabilities in active directory environments.

Exploring the Browser Exploit Framework (BeEF)

- BeEF is a tool designed to exploit some functionality within a browser to launch XSS and injection attacks against a website.
 - The goal is to gain access, gather information, use a proxy, and other utilities for the PenTester.
- One method to gain control is by hooking a browser
 - Connects a browser to another device, usually an attacker's tool or framework, to execute further attacks.
- With BeEF, you can easily 'hook' a browser
 - Using a standard JavaScript file included within the framework.

Exploiting a Browser with BeEF

- On the left side BeEF's main window, you will see a list of Hooked Browsers. Within the section, there are two folders displayed:
 - **Online** informs you that the device is available and awaiting instructions.
 - **Offline** informs you that the device is not ready.
- If you select an IP address of one of the hooked browsers, BeEF will provide some basic information, such as:
 - Web browser, OS, hardware type (if known), location (if known)

Displaying Information in BeEF

- Selecting an IP address is selected will display tabs on the right:
 - Details, Logs and Commands.
- Selecting *Commands* will provide a way to execute a variety of modules used to gather further information about the device.
- Modules include:
 - Type of browser in use, Use as a proxy, Get Internal IP address
- BeEF will indicate which modules will work against the target by using different colors.

Review Activity: Identify Tools

- Compare differences in web testing tools and list some examples
- Discuss how Browser Exploit Framework (BeEF) can be used to test web browsers
- Explain how the team will move through exploiting a browser with BeEF
- Describe how BeEF displays information on hooked browsers

Lesson 13



Summary