ZROC102

Module 3 – Testing Common Web Application Vulnerabilities (OWASP Top 10)

Introduction

Introduction OWASP Top 10 (owasp.org)

Installing Foxy Proxy

Exploring Burp suite

https://www.youtube.com/embed/ouDe5sJ_uC8?origin=https://portswigger.net&rel=0

SQL injection attacks overview

OWASP Top Ten 2021

What is OWASP Top 10?

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.

OWASP Top 10 List for 2021

- A1 Broken Access Control
- A2 Cryptographic Failures
- A3 Injection
- A4 Insecure Design
- A5 Security Misconfiguration
- A6 Vulnerable and Outdated Components
- A7 Identification and Authentication Failures
- A8 Software and Data Integrity Failures
- A9 Security Logging and Monitoring Failures
- A10 Server-Side Request Forgery

WEB SECURITY RISKS CHANGES OVER TIME

OWASP Top Ten 2013 vs 2017

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Image source: https://cipher.com/

Injection

Applications use SQL queries in order to receive, add, change or remove data.
Sending hostile data to an interpreter (e.g. SQLi, LDAP Injection, OS Injection)

String query = "SELECT Cust_No, First_Name FROM customers WHERE Last_Name='Smith'; drop table customers --
"

SQL statements combine code and data

SQLI Demo (DVWA)

Basic Injection

Instructions:

- Input "1" into the text box.
- Click Submit.
- Note, webpage/code is **supposed to** print ID, First name, and Surname to the screen.

Notes(FYI):

- Below is the PHP select statement that we will be exploiting, specifically $id.
  - $getid = "SELECT first_name, last_name FROM users WHERE user_id = '**$id**'";

SQLI

- **Always true scenario**
- **Notes (FYI) :**
  - In this scenario, we are saying display all record that are **false** and all records that are **true.**
    - 1' - Will probably not be equal to anything, and will be false.
    - '0'='0' - Is equal to true, because 0 will always equal 0.
  - Database Statement
    - mysql> SELECT first_name, last_name FROM users WHERE user_id = **'%' or '0'='0';**
- 1' or 0=0 union select null, table_name from information_schema.tables where table_name like 'user%' #

SQLI

Display Database Version

Instructions:

- Input the below text into the User ID Textbox (See Picture).
  - %' or 0=0 union select null, version() #
- Click Submit

Notes(FYI):

- Notice in the last displayed line, 5.1.60 is displayed in the surname.
- This is the version of the mysql database.

SQLI

Display Database User

Instructions:

- Input the below text into the User ID Textbox (See Picture).
  - %' or 0=0 union select null, user() #

Notes(FYI):

- Notice in the last displayed line, root@localhost is displayed in the surname.
- This is the name of the database user that executed the behind the scenes PHP code.

SQLi Injection Prevention

- Parameterize your queries
- a **prepared statement** or **parameterized statement** is a feature used to execute the same or similar database statements repeatedly with high efficiency. Typically used with SQL statements such as queries or updates, the prepared statement takes the form of a template into which certain constant values are substituted during each execution.
- Validate which data can be entered
- Escape special character

Broken Authentication

Broken authentication is an umbrella term for several vulnerabilities that attackers exploit to impersonate legitimate users online.

WEAK SESSION MANAGEMENT

- CREDENTIAL STUFFING

- BRUTE FORCE

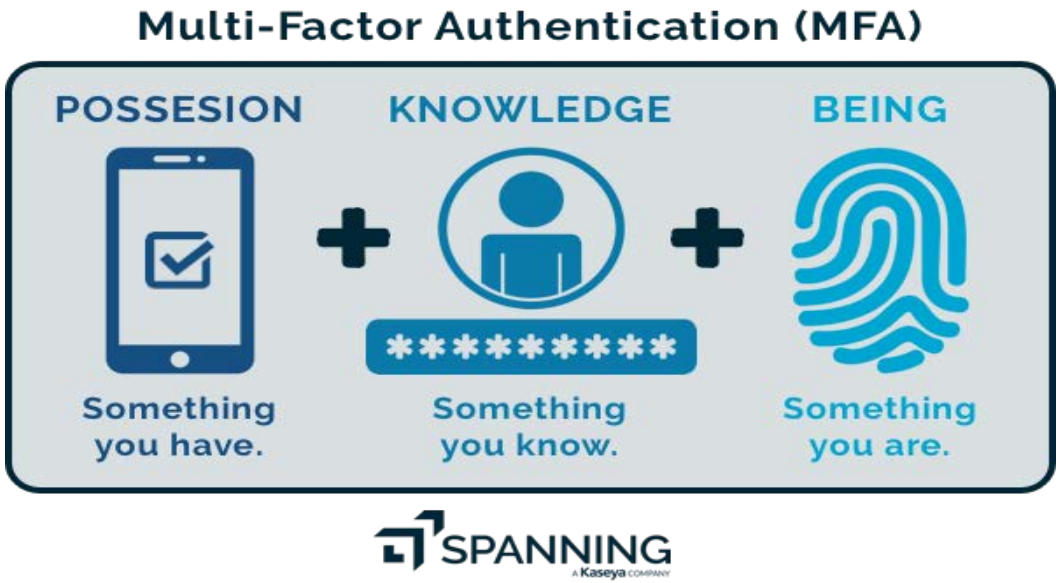
- FORGOTTEN PASSWORD

- NO MULTI-FACTOR AUTHENTICATION

- SESSIONS DON'T EXPIRE

Prevention

- Use MFA (Multi-factor Authentication)



- Enforce strong passwords
- Detect and protect from brute force attacks.

Sensitive Data Exposure

- This attack is related to not taking suitable measures to protect sensitive data at rest or in transit. Eg (using HTTP instead of HTTPS)
- Clear-text data transfer
- Unencrypted storage
- Weak crypto or keys
- Certificates not validated
- Exposing PII or Credit Cards

XML External Entities (XXE)

- This attack covers a much wider more common issue. Many sites which support the uploading of files save the files within the HTTP directory structure which means once uploaded they can be executed via the browser.

- **Solutions**
  - If an upload has a suspected issue quarantine or delete it, don't leave it on the system in the "*uploads*" directory.
  - Rename the uploaded file to something else to make it more difficult for the attacker to find and execute the file.
  - If you need to retrieve the uploaded file via the browser, use a helper function to retrieve the file and serve it. That will give you more control to run additional checks.
  - Make sure your web server has directory browsing disabled. You should not be able to browse to any directory and see the full contents.
  - Make sure your web server does not have directory recursion on. You don't want them to be doing something like this ../../../etc/passwd.

Broken Access Control

This is difficult to demonstrate but possible symptoms of vulnerability are:

- Attacker able to force browse to a page or directory that they should not have permission.

- Attacker able to POST their own custom data to forms and APIs that are not properly verifying the data.

Prevention

- Use proven code or libraries
- Deny access by default
- Log failures and alert
- Rate limit access to resources

Security Misconfiguration

Many services and libraries are installed with default login credentials.

- ✓ Make sure that permissions on files and directories are set correctly and that directory browsing is disabled

- ✓ The web server and programming language(s) should be configured to give away very little in terms of what they are and versions.

Insecure Deserialization

- The easiest way to describe this is a replay attack. The attacker will capture the unencrypted communication between two devices, make changes to the communication, and replay it.
    - For example a PHP application uses PHP object serialization to save a “super” cookie, containing the user’s user ID, role, password hash, and other state. The attacker could alter this cookie to provide them elevated privileges.

Using Components with Known Vulnerabilities

Attackers will often attempt to exploit unpatched flaws. This is actually easier than you may think as there are freely available databases like Exploit DB which contain thousands of exploits and how to use them.

If for example the attacker can figure out that a web application is using PHP 7.4 on Apache 2.4 they just need to search for “php 7.4” and “apache 2.4” in the database and many known exploits are available.

Insufficient Logging & Monitoring

- The bottom line here is if you aren't aware that you are being attacked, you are basically allowing the attacker unlimited time to carry out their attack.
- So what should you be doing?
  - Auditing events like logins, failures, high-value transactions etc.
  - Warnings and errors should generate meaningful log entries.
  - Log suspicious activity (E.g. excessive usage of API calls).
  - Centralize logging… don't store logs locally.
  - Implement a process to action events at certain thresholds in near real-time.
  - Automate security event handling E.g. SOAR (Security Orchestration, Automation, and Response) and Security Information and Event Management (SIEM)

References

- https://owasp.org/www-project-top-ten/
- https://www.cypressdatadefense.com/pdf/OWASP-Top-10-PPT.pdf
- https://www.business2community.com/cybersecurity/why-multi-factor-authentication-mfa-is-a-must-have-in-the-microsoft-world-and-beyond-02245731