Web Application Security Threats and Attacks

Objectives:

At the end of this episode, I will be able to:

Understand what web application security threats and attacks are.

Explain what the OWASP Top 10 Application Security Risks are.

Identify other Web Application Threats that IH&R team members should be familiar with.

External Resources:

Web Application Security Threats and Attacks

What are the OWASP Top 10 Application Security Risks? -

```
▪ A1 – Injection: Injection flaws, such as SQL, command injection, and LDAP
injection occur when attackers send untrusted data to an interpreter as part
of a command or query. The attacker's hostile data can trick the interpreter
into executing unintended commands or accessing data without proper authorization.
```

For example, the following SQL statement:

```
SELECT * FROM tablename WHERE UserID= 2497
```

becomes the following with a simple SQL injection attack:

```
SELECT * FROM tablename WHERE UserID= 2497 OR 1=1
```

The expression "OR 1=1" evaluates to the value "TRUE," often allowing the enumeration of all user ID values from the database.

Attackers carryout SQL injection attacks from the web browser's address bar, form fields, queries, and searches.

SQL injection attacks allow attackers to:

```
▪ Log into the application without supplying valid credentials
▪ Perform queries against data in the database, often even data to which the
application would not normally have access
▪ Modify database contents, or drop the database altogether
▪ Use the trust relationships established between the web application
components to access other databases
```

Command Injection Attacks - Include calls to an operating system over system calls, use of external programs over shell commands, and calls to the backend databases over SQL.

File Injection Attack - Used to exploit "dynamic file include" mechanisms in web applications. Enable attackers to exploit vulnerable scripts on the server to use a remote file instead of a presumably trusted file from the local file system.

Occurs when a user is allowed to supply input for the include command dynamically, and is not properly validated before processing. When a user provides input, the web application passes it into "file include" commands.

Most web application frameworks support file inclusion. The attacker enters a URL that redirects the application to the location of the malicious file, the application executes the file script by calling specific procedures without proper validation.

▪ A2 – Broken Authentication: Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens or to exploit other implementation flaws to assume other users' identities (temporarily or permanently).

▪ A3 – Sensitive Data Exposure: Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII (Personal Identifiable Information). Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

▪ A4 – XML External Entity (XXE): Many older or poorly configured XML processors evaluate external entity references within XML documents. Attackers can use external entities to disclose internal files using the file URI handler, internal SMB file shares on unpatched Windows servers, internal port scanning, remote code execution, and denial of service attacks, such as the Billion Laughs attack.

▪ A5 – Broken Access Control: Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, and change access rights.

▪ A6 – Security Misconfiguration: Security misconfiguration is the most common issue in web security, which is due in part to manual or ad hoc configuration (or not configuring at all), insecure default configurations, misconfigured HTTP headers, error messages containing sensitive information, not patching or upgrading systems, frameworks, dependencies, and components in a timely fashion (or at all).

▪ A7 – Cross-Site Scripting (XSS): Occurs when invalidated input data is included in dynamic content that is sent to a user's web browser for rendering. Attackers inject malicious JavaScript, VBScript, ActiveX, HTML, or Flash for execution on a victim's system by hiding it within legitimate requests. Attackers bypass client-ID security mechanisms and gain access privileges, and then inject malicious scripts into specific web pages. These malicious scripts can even rewrite HTML website content.

▪ A8 – Insecure Deserialization: Occurs when an application receives hostile serialized objects. Insecure deserialization leads to remote code execution. Even if deserialization flaws do not result in remote code execution, attackers can replay, tamper or delete the serialized objects to spoof users, conduct injection attacks, and elevate privileges.

▪ A9 – Using Components with Known Vulnerabilities: Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If attackers exploit a vulnerable component, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

▪ A10 – Insufficient Logging & Monitoring: Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

What about other Web Application Threats? -

▪ Unvalidated Redirects and Forwards - Attackers lure victims and make them click on unvalidated links that appear to be legitimate. Such redirects may attempt to install malware or trick victims into disclosing passwords or other sensitive information. Unsafe forwards may allow access control bypass, leading to:

o Session Fixation Attack
o Security Management Exploits
o Failure to Restrict URL Access
o Malicious File Execution

▪ Cross-Site Request Forgery - An authenticated user is made to perform certain tasks on the web application that an attacker chooses. For example, a user clicking on a particular link sent through an email or chat.

▪ Cookie/Session Poisoning - By changing the information inside a cookie, attackers bypass the authentication process; once they gain control over a network, they can modify its content, use the system for a malicious attack, or steal information from users' systems.

▪ Obfuscation - The most common method of attack obfuscation involves encoding portions of the attack with Unicode, UTF-8, Base64, or URL encoding. Unicode is a method of representing letters, numbers, and special characters to properly display them, regardless of the application or underlying platform.

▪ DMZ Protocol Attacks - The DMZ ("demilitarized zone") is a semi-trusted network zone that separates the untrusted internet from the company's trusted internal network. An attacker, who is able to compromise a system that allows other DMZ protocols, has access to other DMZs and internal systems.