ASSIGNMENT 2

1. SQL Injection

Real-World Case Study

A major financial services provider was breached when attackers used SQL Injection to access sensitive payment data. The exploit allowed unauthorized modifications to the database, leaking millions of records and causing both financial and reputational harm.

Mitigation Strategies

- **Parameterized Queries and Prepared Statements:** Rather than directly concatenating user input into SQL queries, developers should use parameterized queries, ensuring that the database engine treats inputs strictly as data rather than executable code.
- **Input Validation and Sanitization:** Robust validation mechanisms should be in place to restrict inputs to acceptable values (e.g., numeric ranges or specific string formats). Employing whitelisting techniques helps ensure that malicious characters are not processed.
- **Stored Procedures and ORM Frameworks:** Using stored procedures or Object-Relational Mapping (ORM) frameworks can abstract database queries from direct SQL execution, thereby reducing injection risks.
- **Least Privilege Principle:** The database user account used by the web application should have the minimum permissions necessary, limiting the potential damage in case of an injection attack.
- **Regular Code Reviews and Penetration Testing:** Frequent security audits and automated vulnerability scans can help detect and remediate SQL Injection vulnerabilities before attackers can exploit them.

---

2. Cross-Site Scripting (XSS)

Overview

Cross-Site Scripting (XSS) is an attack where attackers inject malicious scripts into webpages. These vulnerabilities happen when an application fails to properly handle user input, allowing the script to run in victims' browsers, hijack sessions, or redirect users to harmful sites

Types of XSS Attacks

- **Stored (Persistent) XSS:** The malicious script is permanently stored on the target server (e.g., in a database or message forum) and is served to users upon each visit.
- **Reflected (Non-Persistent) XSS:** The malicious script is embedded in a URL or form submission and is immediately reflected back by the server in the response.
- **DOM-Based XSS:** The vulnerability is exploited on the client side when the web application’s client-side scripts process user input insecurely.

Real-World Case Study

The "Samy Worm" on MySpace in 2005 is a stored XSS attack where a malicious script sent friend requests to users who viewed the profile, affecting over one million people in a day. It shows how XSS can spread quickly and the need for proper input validation.

Mitigation Strategies

- **Output Encoding:** Convert special characters into HTML entities before rendering user-supplied content. This prevents browsers from interpreting them as executable code.
- **Content Security Policy (CSP):** A robust CSP helps restrict the sources from which scripts, styles, and other resources can be loaded, significantly reducing the impact of XSS.
- **Input Sanitization:** Ensure that inputs are validated for allowed characters and stripped of any potentially dangerous content.
- **Framework Security Features:** Many modern web frameworks include built-in XSS protection; developers should use these features rather than implementing custom solutions.
- **Regular Security Testing:** Automated tools and manual penetration tests should be routinely performed to identify and fix any XSS vulnerabilities.

---

3. Cross-Site Request Forgery (CSRF)

Overview

Cross-Site Request Forgery (CSRF) tricks authenticated users into performing unwanted actions on a trusted website, like transferring funds or changing passwords, by exploiting their browser.

How CSRF Works

Attackers lure users to a malicious site, which sends a fake request using the user's session cookies to a trusted site, making it appear legitimate and causing the trusted site to process it without extra authentication.

Real-World Case Study

High-profile banking incidents have involved attackers exploiting CSRF vulnerabilities to conduct unauthorized fund transfers. By placing hidden forms on malicious websites, attackers trick authenticated bank customers into making transfers. These actions cause financial losses and legal issues for both customers and the banks involved.

Mitigation Strategies

- **Anti-CSRF Tokens:** Unique tokens are generated and included with every form submission. The server verifies the token on each request, ensuring that the request originates from the authenticated user.
- **SameSite Cookies:** Configuring cookies with the “SameSite” attribute restricts how cookies are sent with cross-site requests, reducing the risk of CSRF.

- **Double-Submit Cookies:** This technique involves sending the CSRF token in both a cookie and a request parameter, allowing the server to compare the two values.
- **User Confirmation:** For sensitive transactions, additional verification steps (such as re-entering a password or using multi-factor authentication) can help ensure that the request is genuine.
- **Security Audits:** Regular reviews of the codebase and session management practices can help identify potential CSRF vulnerabilities before they are exploited.

---

4. Session Hijacking

Overview

Session Hijacking is when an attacker takes over a user's active session, impersonating the user to gain unauthorized access. It exploits weaknesses in session management, like insecure cookies or poor session expiration.

How Session Hijacking Works

Session hijacking occurs when an attacker intercepts a user's session ID, often through network sniffing or malware, allowing them to impersonate the user and access their resources and privileges.

Real-World Case Study

Firesheep was a 2010 Firefox extension that highlighted session hijacking vulnerabilities by intercepting session cookies on social media over unsecured Wi-Fi networks. This event raised awareness of public network risks and pushed many websites to improve security by adopting encrypted communications and better session management practices.

Mitigation Strategies

- **Use of HTTPS:** Encrypting data in transit using HTTPS prevents attackers from easily intercepting session IDs on public or unsecured networks.
- **Secure and HttpOnly Flags:** Setting these flags on cookies ensures that cookies are only transmitted over secure channels and are not accessible via client-side scripts.
- **Session Timeout and Re-authentication:** Implementing short session lifetimes and requiring periodic re-authentication reduces the window of opportunity for an attacker.
- **IP and Device Binding:** Tying a session to a user’s IP address or device characteristics can make it more difficult for an attacker to reuse a hijacked session ID.
- **Regular Session Regeneration:** Frequently changing the session ID during a session minimizes the risk of a long-term hijacking.

---

5. Man-in-the-Middle (MITM) Attacks

Overview

MITM attacks involve an attacker intercepting and possibly altering communication between two parties, allowing them to eavesdrop on sensitive data, steal credentials, or modify information in transit.

How MITM Attacks Work

In a typical MITM attack, an attacker intercepts data between a client and server on unsecured networks like public Wi-Fi using methods such as ARP spoofing, DNS spoofing, or rogue access points. They can then record or alter the information without either party knowing.

Real-World Case Study

A popular electronics company had pre-installed software that allowed MITM attacks on encrypted communications. Additionally, rogue Wi-Fi networks in public areas intercepted users' sensitive data. These cases highlight the need for secure networks and strong encryption.

Mitigation Strategies

- **Encryption with TLS/SSL:** Ensuring that all communication is encrypted using Transport Layer Security (TLS) or Secure Sockets Layer (SSL) is the most effective defense against MITM attacks.
- **Certificate Pinning:** This practice involves hardcoding known server certificates or public keys within the application. By comparing the received certificate against a trusted copy, applications can detect and reject suspicious certificates.
- **VPN Usage:** Virtual Private Networks (VPNs) provide an encrypted tunnel for data transmission, making it significantly harder for attackers to intercept communications on public networks.
- **Public Wi-Fi Caution:** Users should avoid using public Wi-Fi for sensitive transactions. When necessary, a VPN or mobile data connection is preferable.
- **Regular Network Monitoring:** Continuous monitoring for unusual network traffic or unexpected certificate changes can help identify and mitigate MITM attempts in real time.