Theory and Research

1. OWASP Top 10 Vulnerabilities

The OWASP Top 10 highlights the most critical security risks to web applications. Here are the details in simple terms:

- **SQL Injection:** Attackers trick a website into running harmful commands in the database. This can give them access to important information or allow them to change or delete data. For example, if a login form doesn't validate user input, attackers can inject SQL commands to bypass authentication.
- **Cross-Site Scripting (XSS):** Malicious code is added to web pages so attackers can steal information like login details or make users visit harmful websites. For instance, if a comment section allows users to submit scripts, attackers can insert malicious JavaScript to steal other users' cookies.
- **Cross-Site Request Forgery (CSRF):** This tricks users into doing actions they didn't intend to, like transferring money or changing passwords, while they are logged in. Attackers often embed malicious links in emails or websites to exploit this vulnerability.
- **Security Misconfiguration:** Happens when websites are not set up correctly, such as using default settings, having unnecessary features enabled, or forgetting to apply security headers. For example, leaving debugging information visible can give attackers clues about vulnerabilities.
- **Broken Authentication:** Weak passwords or improper handling of login sessions allow attackers to pretend to be someone else. This might include flaws like storing passwords in plain text or not invalidating sessions after logout.
- **Sensitive Data Exposure:** Data like passwords, credit card numbers, or personal details are not properly secured, making it easy for attackers to steal them. For example, transmitting sensitive data over HTTP instead of HTTPS exposes it to interception.
- **Insecure Deserialization:** When websites improperly process data sent by users, attackers can take control of the application. This often happens when objects are deserialized without checking the content, allowing attackers to execute malicious code.
- **Components with Known Vulnerabilities:** Using old software with known flaws can make the website an easy target for attackers. For instance, running outdated versions of plugins or libraries can expose the system to exploitation.
- **Insufficient Logging & Monitoring:** If a website doesn't keep proper logs or monitor for unusual activity, it may take a long time to detect and respond to attacks. For example, not recording failed login attempts can let attackers brute-force accounts undetected.
- **XML External Entity (XXE):** Attackers abuse features in older systems to read sensitive information or run harmful commands through specially crafted XML files. For example, poorly configured XML parsers can allow attackers to access server files.

2. Website Security Measures

Websites use these methods to protect themselves:

- **CAPTCHA:** Helps ensure only real people (not bots) can access or use certain parts of the website, preventing automated attacks. For example, CAPTCHAs are often used on login and registration forms to block spam bots.
- **HTTPS:** Encrypts communication between the user and the website so hackers cannot see or steal sensitive information. For instance, HTTPS ensures that payment details entered on an e-commerce site remain secure during transmission.
- **Content Security Policy (CSP):** Restricts where a website can load content from, stopping harmful scripts from running. For example, a CSP can block malicious scripts hosted on unauthorized domains.
- **Firewalls:** Act like a shield, blocking harmful traffic and preventing attacks on the website. A Web Application Firewall (WAF) can specifically filter and monitor HTTP requests to detect and block malicious activity.
- **Two-Factor Authentication (2FA):** Adds extra protection by requiring users to verify their identity using a second method, like a code sent to their phone. For instance, even if someone steals a password, they cannot log in without the second factor.
- **Regular Security Patches:** Keeping all software up to date fixes security issues and reduces the chance of attacks. For example, updating a CMS like WordPress ensures that vulnerabilities in older versions are patched.
- **Input Validation:** Checks and cleans any information users enter to stop harmful data from being used by the website. For example, validating email input ensures it conforms to expected formats and doesn't include malicious code.
- **Secure Session Management:** Protects user sessions with tools like encrypted cookies and automatic logout after inactivity. For instance, setting session cookies with the Secure and HTTPOnly flags prevents attackers from stealing them.
- **Intrusion Detection Systems (IDS**): Monitors website activity for signs of attacks or unusual behavior and alerts administrators. For example, an IDS can flag repeated failed login attempts as a potential brute-force attack.
- **Data Encryption:** Converts sensitive data into a secure format so it remains safe, even if attackers get access to it. For example, encrypting user passwords using strong hashing algorithms like bcrypt protects them even if the database is breached.