Assignment 1:

1. **Injection Attacks:**

Injection attacks occur when malicious code is inserted into an application, often through user inputs. The most common type is SQL Injection, where attackers manipulate input fields to gain unauthorized access to a database. Another form is Cross-Site Scripting (XSS), which involves injecting malicious scripts into web pages, potentially compromising user data or session information.

Exploitation: Hackers can input malicious code, such as SQL queries or JavaScript, into an application's input fields. If the application doesn't properly validate and sanitize this input, the malicious code can execute within the application's context. For SQL Injection, this can lead to unauthorized database access or even complete control over the database.

Prevention: Developers should use parameterized queries or prepared statements to ensure that user input is treated as data, not code. Additionally, input validation and output encoding can help sanitize user inputs and protect against XSS attacks.

1. **Broken Authentication:**

Broken authentication vulnerabilities occur when an application fails to properly manage user authentication and session management. This can lead to unauthorized access, identity theft, or account hijacking. Weak password policies, insecure session management, and lack of multi-factor authentication are common culprits in such vulnerabilities.

Exploitation: Attackers can exploit weak authentication mechanisms or session management flaws to gain unauthorized access. This may involve brute force attacks to guess passwords, session fixation, or session hijacking to impersonate legitimate users.

Prevention: Developers should implement strong password policies, enforce session timeouts, and use secure session management practices. Multi-factor authentication adds an extra layer of security by requiring users to verify their identity in multiple ways.

1. **Sensitive Data Exposure:**

Sensitive data exposure happens when an application doesn't adequately protect sensitive information like credit card details or personal identification numbers (PINs). This data can be exposed through inadequate encryption, improper storage, or insecure transmission methods. Cybercriminals can exploit these weaknesses to steal valuable data.

Exploitation: When sensitive data isn't properly protected, hackers can intercept it during transmission or exploit vulnerabilities in the application to gain access to stored data. This information can be sold on the black market or used for identity theft.

Prevention: Data should be encrypted both in transit and at rest. Employ secure key management and never store sensitive information like credit card numbers or Social Security numbers unless absolutely necessary. Regularly review and audit access controls to ensure only authorized users can access sensitive data.

1. **XML External Entity (XXE) Attacks:**

XXE attacks target applications that parse XML input from untrusted sources. Attackers can leverage these vulnerabilities to access internal files and resources, potentially leading to data leaks or server compromise. Proper input validation and disabling external entity references can mitigate this risk.

Exploitation: By injecting malicious XML entities, attackers can manipulate how an application parses XML data, potentially leading to data exfiltration or server-side request forgery (SSRF) attacks. These attacks can reveal internal files and resources or launch attacks on other systems.

Prevention: Disable external entity references, validate and sanitize XML input, and use a strict XML parser configuration that doesn't allow the inclusion of external entities. Implement proper error handling to avoid revealing sensitive information.

1. **Security Misconfigurations:**

Security misconfigurations result from improper setup or configuration of an application or its components. This can include leaving default passwords, overly permissive access controls, or exposing sensitive information in error messages. Cybercriminals often scan for such weaknesses to gain unauthorized access or exploit vulnerabilities.

Exploitation: Hackers actively search for misconfigured applications. They might exploit default credentials, access sensitive files left exposed, or leverage overly permissive settings. Error messages that reveal configuration details can also aid attackers.

Prevention: Regularly audit application configurations, apply the principle of least privilege to user access, and disable unnecessary services and features. Custom error messages should avoid exposing sensitive information and simply display a generic error message to users.



