M. Eshwar

21BLC1370

Assignment -1

1. Injection

Injection occurs when an attacker exploits insecure code to insert (or inject) their own code into a program. Because the program is unable to determine code inserted in this way from its own code, attackers are able to use injection attacks to access secure areas and confidential information as though they are trusted users. Examples of injection include SQL injections, command injections, CRLF injections, and LDAP injections.

Application security testing can reveal injection flaws and suggest remediation techniques such as stripping special characters from user input or writing parameterized SQL queries.

2. Broken Authentication

Incorrectly implemented authentication and session management calls can be a huge security risk. If attackers notice these vulnerabilities, they may be able to easily assume legitimate users' identities.

Multifactor authentication is one way to mitigate broken authentication. Implement DAST and SCA scans to detect and remove issues with implementation errors before code is deployed.

3. Sensitive Data Exposure

APIs, which allow developers to connect their application to third-party services like Google Maps, are great time-savers. However, some APIs rely on insecure data transmission methods, which attackers can exploit to gain access to usernames, passwords, and other sensitive information.

Data encryption, tokenization, proper key management, and disabling response caching can all help reduce the risk of sensitive data exposure.

4. XML External Entities

This risk occurs when attackers are able to upload or include hostile XML content due to insecure code, integrations, or dependencies. An SCA scan can find risks in third-party components with known vulnerabilities and will warn you about them. Disabling XML external entity processing also reduces the likelihood of an XML entity attack.

5. Broken Access Control

If authentication and access restriction are not properly implemented, it's easy for attackers to take whatever they want. With broken access control flaws, unauthenticated or unauthorized users may have access to sensitive files and systems, or even user privilege settings.

Configuration errors and insecure access control practices are hard to detect as automated processes cannot always test for them. Penetration testing can detect missing authentication, but other methods must be used to determine configuration problems. Weak access controls and issues with credentials management are preventable with secure coding practices, as well as preventative measures like locking down administrative accounts and controls and using multi-factor authentication.