Stuti Maitra Sarkar
21BCI0126

10 web application vulnerabilities:

1. Insecure File Uploads:

Insecure file uploads are a serious vulnerability in web applications that allow users to upload files without adequate validation and security checks. Attackers can exploit this vulnerability by uploading malicious files, which can lead to a range of security risks. These risks may include compromising the web server, spreading malware to other users who download the uploaded files, and even executing arbitrary code on the server. It is essential for web developers to implement strong security measures to validate and sanitize uploaded files to prevent these potential threats.

2. XML External Entity (XXE) Attacks:

XML External Entity (XXE) vulnerabilities are a common issue in web applications that process XML input from untrusted sources. Attackers can exploit these vulnerabilities by crafting malicious XML entities that can lead to various security risks. This can include reading internal files, initiating denial-of-service attacks by exhausting system resources, and in some cases, executing remote code. Protecting against XXE attacks requires proper input validation and secure XML processing practices.

3. Server-Side Template Injection (SSTI):

Server-Side Template Injection (SSTI) vulnerabilities occur when user input is directly embedded in server-side templates. Attackers can take advantage of this vulnerability by injecting malicious code into templates, potentially leading to arbitrary code execution on the server. SSTI vulnerabilities can be particularly dangerous as they can result in data breaches or even full server compromise. To prevent SSTI attacks, developers should carefully validate and sanitize user input and use secure template engines.

4. Broken Authentication and Session Management:

Broken authentication and session management vulnerabilities are a recurring issue in web applications. These vulnerabilities arise when developers fail to implement adequate security measures to protect user authentication and session data. Attackers can exploit these weaknesses to hijack user sessions, impersonate legitimate users, or gain unauthorized access to sensitive resources. Protecting against these vulnerabilities requires proper implementation of secure session management, strong authentication mechanisms, and regular security checks.

5. Insecure Deserialization:

Insecure deserialization vulnerabilities can be a critical security risk in web applications. When attackers exploit these vulnerabilities, they can manipulate data structures used for deserialization, potentially leading to remote code execution. Such attacks can result in compromised systems, data

breaches, and unauthorized access to sensitive resources. Developers should use secure deserialization practices and be cautious when handling serialized data from untrusted sources.

6. Security Misconfiguration:

Security misconfiguration is a prevalent web application vulnerability that occurs when applications are not properly configured. Such misconfigurations can expose sensitive information or grant access to unauthorized users. Attackers seek to exploit these misconfigurations to gather valuable information or gain access to resources they should not have access to. To mitigate this risk, developers should perform regular security assessments, maintain proper configurations, and adhere to security best practices.

7. HTTP Parameter Pollution (HPP):

HTTP Parameter Pollution (HPP) is a web application vulnerability that occurs when attackers manipulate HTTP request parameters to deceive the application. This can lead to unexpected behavior, potentially disclosing sensitive data or enabling unauthorized actions. By skillfully altering parameter values, attackers aim to exploit the application's logic and compromise security. It is essential for developers to implement robust input validation and sanitization to protect against HPP attacks.

8. Cross-Site Request Forgery (CSRF):

Cross-Site Request Forgery (CSRF) vulnerabilities are a type of web security issue that can deceive authenticated users into performing unintended actions without their knowledge. This can lead to unexpected state changes, unauthorized data alterations, and unintended actions within a web application. Attackers aim to trick users into executing actions that benefit the attacker. Preventing CSRF attacks requires implementing anti-CSRF tokens, which verify the authenticity of the user's requests.

9. Content Security Policy (CSP) Bypass:

Bypassing Content Security Policy (CSP) controls is a significant concern for web applications. Attackers look for ways to circumvent CSP rules, which are intended to prevent cross-site scripting (XSS) attacks and protect web content's integrity. If attackers successfully bypass CSP, they can execute scripts that may lead to XSS vulnerabilities or other malicious actions. Proper CSP implementation and regular monitoring are crucial to defend against CSP bypass attacks.

10. Server-Side Request Forgery (SSRF):

Server-Side Request Forgery (SSRF) vulnerabilities enable attackers to make requests from the web server to internal or external resources. Attackers exploit SSRF to access sensitive information,

disrupt services, and potentially launch further attacks within a network. SSRF is a complex and often subtle vulnerability that requires thorough input validation and security measures to mitigate.