Top 20 Application Vulnerabilities: Definitions and Mitigation Steps

Prepared by HANIM EKEN

https://ie.linkedin.com/in/hanimeken

1. Injection Flaws

Definition: Injection flaws occur when untrusted data is sent to an interpreter as part of a command or query. This allows attackers to execute malicious commands or access unauthorized data.
Mitigation Steps:
Use parameterized queries or prepared statements in databases to prevent SQL injection.
Implement input validation and sanitization to filter out malicious input.
Employ least privilege principles to restrict access to sensitive resources.

2. Broken Authentication

Definition: Broken authentication vulnerabilities arise when authentication mechanisms are improperly implemented, allowing attackers to compromise user accounts, passwords, or session tokens.
Mitigation Steps:
Use strong password policies and multi-factor authentication.
Implement secure session management techniques, such as session timeouts and secure cookies.
Regularly review and update authentication mechanisms to address emerging threats.

3. Cross-Site Scripting (XSS)

Definition: XSS vulnerabilities enable attackers to inject malicious scripts into web pages viewed by other users. These scripts can steal session cookies, redirect users to malicious sites, or deface websites.
Mitigation Steps:
Encode user input to prevent script injection.
Implement Content Security Policy (CSP) headers to restrict the execution of scripts.
Conduct regular security testing, including static and dynamic code analysis.

4. Broken Access Control

Definition: Broken access control vulnerabilities occur when developers fail to properly enforce
restrictions on what authenticated users are allowed to do, allowing unauthorized access to sensitive data or functionality.
Mitigation Steps:
Implement role-based access control (RBAC) to enforce least privilege.
Conduct regular access control reviews and audits to identify and remediate misconfigurations.
Use access control lists (ACLs) and enforce proper authorization checks.

5. Security Misconfiguration

Definition: Security misconfiguration vulnerabilities arise when systems are configured insecurely, leaving them vulnerable to exploitation. This includes default settings, unnecessary features enabled, or outdated software.
Mitigation Steps:
Follow secure configuration guides provided by vendors.

Regularly patch and update software to address known vulnerabilities.
Implement automated configuration management tools to enforce security policies.

6. Sensitive Data Exposure

Definition: Sensitive data exposure vulnerabilities occur when sensitive information, such as passwords or credit card numbers, is not properly protected, allowing attackers to access or steal the data.
Mitigation Steps:
Encrypt sensitive data at rest and in transit using strong cryptographic algorithms.
Implement data masking and tokenization techniques to minimize exposure.
Limit access to sensitive data based on the principle of least privilege.

7. Insufficient Logging and Monitoring

Definition: Insufficient logging and monitoring vulnerabilities occur when systems do not properly log security-relevant events or do not monitor logs for suspicious activity, making it difficult to detect and respond to security incidents.
Mitigation Steps:
Implement centralized logging and monitoring solutions.
Define clear logging and alerting policies for security events.
Regularly review and analyze logs for signs of anomalous behavior.

8. Insecure Deserialization

Definition: Insecure deserialization vulnerabilities arise when untrusted data is deserialized by an application, leading to arbitrary code execution or other malicious activities.
Mitigation Steps:
Use serialization libraries with built-in security features.
Implement integrity checks and digital signatures for serialized objects.
Restrict deserialization to trusted sources and avoid deserializing untrusted data.

9. XML External Entity (XXE) Injection

Definition: XXE injection vulnerabilities occur when an application parses XML input insecurely, allowing attackers to read sensitive files, execute arbitrary code, or perform other malicious actions.
Mitigation Steps:
Disable external entity resolution in XML parsers.
Use whitelists to restrict the types of XML entities allowed.
Validate and sanitize XML input to prevent XXE injection attacks.

10. Broken Function Level Authorization

Definition: Broken function level authorization vulnerabilities arise when applications fail to properly enforce access controls on individual functions or endpoints, allowing unauthorized access to restricted functionality.
Mitigation Steps:
Implement access controls at the function or method level.

Conduct thorough access control testing to identify and remediate vulnerabilities.
Use role-based access control (RBAC) to enforce fine-grained access controls.

11. Insecure Direct Object References (IDOR)

Definition: Insecure direct object reference vulnerabilities occur when applications expose internal objects such as files or database records without proper authorization, allowing attackers to access sensitive data.
Mitigation Steps:
Use indirect references or object identifiers to access sensitive resources.
Implement access controls and authorization checks to validate user permissions.
Encrypt sensitive object references to prevent enumeration attacks.

12. Cross-Site Request Forgery (CSRF)

Definition: CSRF vulnerabilities occur when attackers trick authenticated users into unknowingly submitting malicious requests, leading to unauthorized actions being performed on behalf of the victim.
Mitigation Steps:
Use anti-CSRF tokens to validate the origin of requests.
Implement SameSite cookie attributes to prevent CSRF attacks.
Require re-authentication for sensitive actions to mitigate CSRF risk.

13. Using Components with Known Vulnerabilities

Definition: This vulnerability arises when applications use outdated or vulnerable third-party components, libraries, or frameworks, making them susceptible to exploitation by attackers.
Mitigation Steps:
Regularly update and patch third-party components to address known vulnerabilities.
Monitor vendor security advisories and apply security patches promptly.
Use dependency management tools to track and manage software dependencies.

14. Invalidated Redirects and Forwards

Definition: Invalidated redirects and forwards vulnerabilities occur when applications redirect or forward users to untrusted destinations based on user-supplied input, potentially leading to phishing attacks or other malicious activities.
Mitigation Steps:
Avoid using user-controlled input in redirect or forward URLs.
Implement whitelist-based validation to restrict allowed redirect destinations.
Display warning messages to users when redirecting to external URLs.

15. Remote Code Execution (RCE)

Definition: RCE vulnerabilities allow attackers to execute arbitrary code on a targeted system, enabling them to take control of the system, steal data, or launch further attacks.
Mitigation Steps:
Use secure coding practices to prevent code injection vulnerabilities.
Implement input validation and output encoding to sanitize user input.

Employ runtime protections such as sandboxing or containerization to mitigate RCE risk.

16. Insecure File Upload

Definition: Insecure file upload vulnerabilities occur when applications allow users to upload files without proper validation, leading to the execution of malicious code or the uploading of malware-infected files.
Mitigation Steps:
Validate file types and content before accepting uploads.
Store uploaded files in a secure location with restricted access permissions.
Implement malware scanning and content filtering for uploaded files.

17. Server-Side Request Forgery (SSRF)

Definition: SSRF vulnerabilities allow attackers to make unauthorized requests from the server to internal or external resources, potentially exposing sensitive data or services to exploitation.
Mitigation Steps:
Use whitelists to restrict allowed request destinations.
Implement URL validation and filtering to block malicious requests.
Isolate server-side processes and limit network access to internal resources.

18. Improper Input Validation

Definition: Improper input validation vulnerabilities occur when applications fail to properly validate and sanitize user input, allowing attackers to inject malicious code or bypass security controls.
Mitigation Steps:
Use input validation routines and input sanitization libraries to filter out malicious input.
Apply regular expression checks and input length restrictions to prevent injection attacks.
Conduct security testing, including fuzz testing and penetration testing, to identify input validation vulnerabilities.

19. Cryptographic Issues

Definition: Cryptographic issues arise when applications use weak or insecure cryptographic algorithms, improper key management, or other cryptographic flaws, making them susceptible to attacks such as brute force or encryption bypass.
Mitigation Steps:
Use well-established cryptographic algorithms and protocols.
Generate strong, random cryptographic keys and protect them from unauthorized access.
Regularly review cryptographic implementations for compliance with security best practices.

20. Insecure API

Definition: Insecure API vulnerabilities occur when APIs are not properly secured, allowing attackers to access sensitive data, execute unauthorized actions, or manipulate the application's behavior.
Mitigation Steps:

Implement authentication and authorization mechanisms for API access.
Use secure communication protocols such as HTTPS/TLS to encrypt API traffic.
Apply input validation and output encoding to prevent injection and XSS attacks on API endpoints.

HANIM EKEN

https://ie.linkedin.com/in/hanimeken