Friday, July 7, 2023

Remember to GOOGLE for Payloads, BAD Characters, and for additional enumeration!

Title

- Information Gathering:
    - Identify the target's scope, including URLs, subdomains, and relevant technologies.
    - Conduct reconnaissance to gather information from public sources.
    - Perform subdomain enumeration and identify any misconfigurations.
- Authentication and Session Management:
    - Test for weak or predictable passwords.
    - Check for username enumeration.
    - Test for session fixation, session hijacking, or session-related vulnerabilities.
    - Verify if multi-factor authentication (MFA) is implemented correctly.
    - Test for bypassing authentication mechanisms.
    - Check for logout functionality and session termination.
- Authorization and Access Control:
    - Test for privilege escalation vulnerabilities.
    - Check for insecure direct object references (IDOR).
    - Verify if access control rules are enforced correctly.
    - Test for vertical and horizontal privilege escalation.
    - Ensure sensitive data is protected based on user roles and permissions.
- Input Validation and Security:
    - Test for common web vulnerabilities such as cross-site scripting (XSS) and SQL injection.
    - Check for insecure file uploads and file inclusion vulnerabilities.

    - Test for remote code execution (RCE) and command injection vulnerabilities.
    - Validate input fields for input length and data types.
    - Check for HTTP parameter pollution (HPP) vulnerabilities.
- Error Handling and Information Leakage:
    - Test for error-based vulnerabilities and improper error handling.
    - Verify if sensitive information is being leaked in error messages or logs.
    - Check for directory listing and file disclosure vulnerabilities.
    - Test for server-side request forgery (SSRF) and XML external entity (XXE) vulnerabilities.
- Security Headers and Configuration:
    - Check for secure communication protocols (TLS/SSL).
    - Verify if Content Security Policy (CSP) and Cross-Origin Resource Sharing (CORS) are properly implemented.
    - Test for HTTP security headers such as Strict-Transport-Security (HSTS) and X-XSS-Protection.
    - Ensure secure cookie configurations (Secure flag, HttpOnly flag, etc.).
- Business Logic and Application Flaws:
    - Test for logical flaws, such as bypassing business rules or authorization checks.
    - Look for privilege escalation opportunities by manipulating business processes.
    - Test for insecure direct object references (IDOR) and data exposure.
- API and Web Services:
    - Test for API vulnerabilities, such as insecure API endpoints or excessive data exposure.
    - Verify if proper authentication and access controls are in place.
    - Check for API rate limiting and throttling mechanisms.

        - Test for API injection vulnerabilities and insecure deserialization.
- Miscellaneous:
    - Test for server-side vulnerabilities like server misconfigurations or outdated software.
    - Check for clickjacking and cross-site request forgery (CSRF) vulnerabilities.
    - Test for DOM-based vulnerabilities and client-side security issues.
    - Verify if security controls are in place for sensitive operations (password resets, account deletion, etc.).

1: Review application input points and assess their vulnerability to XSS attacks.

- Test for reflected XSS by injecting script tags and malicious payloads into input fields and parameters.
- Test for stored XSS by submitting data that gets stored and later displayed to users.
- Check if proper output encoding is applied to user-generated or dynamic content.
- Evaluate the effectiveness of content security policies (CSP) and other XSS mitigation techniques.

2: Improper Access Control - Generic

- Identify sensitive functionalities and resources that require proper access controls.
- Test for insecure direct object references (IDOR) by manipulating identifiers or parameters.
- Verify if access controls are enforced consistently and adequately for different user roles.
- Test for privilege escalation opportunities by bypassing or manipulating authorization mechanisms.
- Check if secure session management is implemented to prevent unauthorized access.

3: Information Disclosure

- Review error handling mechanisms and test for potential information disclosure in error messages.
- Check for sensitive information exposed in server responses, headers, or log files.
- Assess directory listing vulnerabilities that may reveal directory or file structures.
- Verify if secure configuration settings are in place to prevent unnecessary information disclosure.
- Test for potential data leakage through insecure data transmission channels.

4: Server-Side Request Forgery (SSRF)

- Identify user-controlled input points that can be used to make arbitrary requests.
- Test for SSRF by providing different URLs and verifying if the server fetches and responds with their content.
- Check for blind SSRF by analyzing timing or error-based responses.
- Evaluate the effectiveness of server-side input validation and whitelisting.
- Verify if SSRF protection mechanisms, such as URL whitelisting or blacklisting, are implemented.

5: Insecure Direct Object Reference (IDOR)

- Identify direct object references in the application's URLs, parameters, or hidden fields.
- Test for insecure direct object references by manipulating identifiers or parameters.
- Verify if proper access controls and authorization checks are implemented for sensitive resources.
- Assess the effectiveness of random or encrypted identifiers to prevent IDOR vulnerabilities.
- Check if access to sensitive data is enforced based on user roles and permissions.

6: Privilege Escalation

- Identify areas where user privileges can be escalated, such as changing roles or manipulating parameters.
- Test for vertical privilege escalation by attempting to access higher-level functionalities or data.
- Test for horizontal privilege escalation by manipulating parameters or session data.
- Verify if access controls and authorization checks are consistently applied across the application.
- Assess the effectiveness of session management to prevent unauthorized privilege escalation.

7: SQL Injection

- Identify user input points that interact with the database.
- Test for SQL injection by injecting malicious SQL statements or payloads.
- Use techniques like UNION-based, time-based, or error-based attacks to assess vulnerability.
- Verify if proper input validation and parameterization techniques are implemented.
- Assess the effectiveness of server-side input sanitization and output encoding mechanisms.

8: Improper Authentication - Generic

- Review the authentication mechanism and its implementation.
- Test for weak or predictable passwords, including default or common credentials.
- Assess the effectiveness of password complexity requirements and password reset functionality.
- Test for session-related vulnerabilities, such as session fixation or session hijacking.
- Verify if multi-factor authentication (MFA) is implemented and properly enforced.

9: Code Injection

- Identify user input points that interact with the application's code execution.
- Test for code injection vulnerabilities, such as operating system (OS) command injection or remote code execution (RCE).
- Verify if proper input validation and sanitization techniques are implemented.
- Assess the effectiveness of output encoding and secure coding practices.
- Check if server-side security measures like executing code within a restricted environment are implemented.

10: Cross-Site Request Forgery (CSRF)

- Identify state-changing requests that can be forged.
- Test for CSRF vulnerabilities by constructing malicious requests and tricking authenticated users into executing them.
- Verify if anti-CSRF tokens are used and validated properly.
- Assess the effectiveness of referrer checks or other CSRF protection mechanisms.
- Test for CSRF in different contexts, such as forms, AJAX requests, or API endpoints.