OWASP ZAP Vulnerability Report

1. Absence of Anti-CSRF Tokens

Risk: Medium (Low)
Description: <p>No Anti-CSRF tokens were found in a HTML submission form.</p><p>A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.</p><p></p><p>CSRF attacks are effective in a number of situations, including:</p><p> * The victim has an active session on the target site.</p><p> * The victim is authenticated via HTTP auth on the target site.</p><p> * The victim is on the same local network as the target site.</p><p></p><p>CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.</p>
Solution: <p>Phase: Architecture and Design</p><p>Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.</p><p>For example, use anti-CSRF packages such as the OWASP CSRFGuard.</p><p></p><p>Phase: Implementation</p><p>Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.</p><p></p><p>Phase: Architecture and Design</p><p>Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).</p><p>Note that this can be bypassed using XSS.</p><p></p><p>Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.</p><p>Note that this can be bypassed using XSS.</p><p></p><p>Use the ESAPI Session Management control.</p><p>This control includes a component for CSRF.</p><p></p><p>Do not use the GET method for any request that triggers a state change.</p><p></p><p>Phase: Implementation</p><p>Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.</p>

2. Content Security Policy (CSP) Header Not Set

Risk: Medium (High)
Description: <p>Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.</p>
Solution: <p>Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.</p>

3. Missing Anti-clickjacking Header

Risk: Medium (Medium)
Description: <p>The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.</p>
Solution: <p>Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app.</p><p>If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.</p>

4. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Risk: Low (Medium)
Description: <p>The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.</p>
Solution: <p>Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.</p>

5. Server Leaks Version Information via "Server" HTTP Response Header Field

Risk: Low (High)
Description: <p>The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.</p>
Solution: <p>Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.</p>

6. X-Content-Type-Options Header Missing

Risk: Low (Medium)
Description: <p>The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.</p>
Solution: <p>Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.</p><p>If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.</p>

7. Authentication Request Identified

Risk: Informational (Low)
Description: <p>The given request has been identified as an authentication request. The 'Other Info' field contains a set of key=value lines which identify any relevant fields. If the request is in a context which has an Authentication Method set to "Auto-Detect" then this rule will change the authentication to match the request identified.</p>
Solution: <p>This is an informational alert rather than a vulnerability and so there is nothing to fix.</p>

8. Base64 Disclosure

Risk: Informational (Medium)
Description: <p>Base64 encoded data was disclosed by the application/web server. Note: in the interests of performance not all base64 strings in the response were analyzed individually, the entire response should be looked at by the analyst/security team/developer(s).</p>
Solution: <p>Manually confirm that the Base64 data does not leak sensitive information, and that the data cannot be aggregated/used to exploit other vulnerabilities.</p>

9. Charset Mismatch (Header Versus Meta Content-Type Charset)

Risk: Informational (Low)
Description: <p>This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set.</p><p></p><p>An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.</p>

Solution: <p>Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.</p>

10. Modern Web Application

Risk: Informational (Medium)
Description: <p>The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.</p>
Solution: <p>This is an informational alert and so no changes are required.</p>

11. Sec-Fetch-Dest Header is Missing

Risk: Informational (High)
Description: <p>Specifies how and where the data would be used. For instance, if the value is audio, then the requested resource must be audio data and not any other type of resource.</p>
Solution: <p>Ensure that Sec-Fetch-Dest header is included in request headers.</p>

12. Sec-Fetch-Mode Header is Missing

Risk: Informational (High)
Description: <p>Allows to differentiate between requests for navigating between HTML pages and requests for loading resources like images, audio etc.</p>
Solution: <p>Ensure that Sec-Fetch-Mode header is included in request headers.</p>

13. Sec-Fetch-Site Header is Missing

Risk: Informational (High)
Description: <p>Specifies the relationship between request initiator's origin and target's origin.</p>
Solution: <p>Ensure that Sec-Fetch-Site header is included in request headers.</p>

14. Sec-Fetch-User Header is Missing

Risk: Informational (High)
Description: <p>Specifies if a navigation request was initiated by a user.</p>
Solution: <p>Ensure that Sec-Fetch-User header is included in user initiated requests.</p>

15. User Controllable HTML Element Attribute (Potential XSS)

Risk: Informational (Low)
Description: <p>This check looks at user-supplied input in query string parameters and POST data to identify where certain HTML attribute values might be controlled. This provides hot-spot detection for XSS (cross-site scripting) that will require further review by a security analyst to determine exploitability.</p>

Solution: <p>Validate all input and sanitize output it before writing to any HTML attributes.</p>

Report generated on: 4/5/2025, 4:11:48 pm