future interns

Vulnerability Assessment Report

Target Website: **https://Testfire.net**

Assessment Type: **Vulnerability Assessment**

Scope: **Read-Only** (No exploitation, no data modification)

Total Vulnerabilities Identified: 09

Executive Summary

A vulnerability assessment was conducted on the target web application to identify security misconfigurations and potential weaknesses. The assessment identified **09 vulnerabilities**, primarily related to **missing security headers, weak cookie attributes, and information disclosure**. No critical or high-risk vulnerabilities were detected; however, multiple **medium-risk issues** could be chained together to perform attacks such as **CSRF, clickjacking, session hijacking ,Cross-Site Request Forgery (CSRF), Clickjacking, Cross-Site Scripting (XSS), session abuse, and reconnaissance attacks.**

Vulnerability Breakdown

Medium Risk Vulnerabilities

1.Absence of Anti-CSRF Tokens

Summary

No Anti-CSRF tokens were found in a HTML submission form. 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.

Impact:
An attacker could craft a malicious webpage that forces an authenticated victim to perform unintended actions, such as changing account settings or submitting unauthorized requests.
Recommendation:
Implement unique, unpredictable Anti-CSRF tokens for all state-changing requests

Solution

Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness **easier to avoid.** For example, use anti-CSRF packages such as the OWASP CSRFGuard. Phase: Implementation Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script. Phase: Architecture and Design 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). Note that this can be bypassed using XSS. 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. Note that this can be bypassed using XSS. Use the ESAPI Session Management control. This control includes a component for CSRF. Do not use the GET method for any request that triggers a state change. Phase: Implementation 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.

Other Info

No known Anti-CSRF token [token, csrfToken, csrf-token] was found in the following HTML form: [Form 1: "name"].

References

- https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
- https://cwe.mitre.org/data/definitions/352.html

Details	
Alert ID	10202
Alert Type	Passive
Status	release
Risk	Medium
CWE	352
WASC	9
Technologies Targeted	All
Tags	CWE-352 OWASP_2017_A05 OWASP_2021_A01 POLICY_DEV_STD POLICY_PENTEST POLICY_QA_STD SYSTEMIC WSTG-V42-SESS-05
More Info	Scan Rule Help

2.Content Security Policy (CSP) Header Not Set

Summary

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.

- **Impact:** Script injection, data theft
- **Recommendation:** Define a strict CSP header allowing trusted sources only.

Solution

Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

References

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP
- https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
- https://www.w3.org/TR/CSP/
- https://w3c.github.io/webappsec-csp/
- https://web.dev/articles/csp
- https://caniuse.com/#feat=contentsecuritypolicy
- https://content-security-policy.com/

Details	
Alert ID	10038-1
Alert Type	Passive
Status	release
Risk	Medium
CWE	693
WASC	15
Technologies Targeted	All
Tags	CWE-693 OWASP_2017_A06 OWASP_2021_A05 POLICY_PENTEST POLICY_QA_STD SYSTEMIC
More Info	Scan Rule Help

3.Missing Anti-Clickjacking Header

Summary
The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Impact: Clickjacking attacks
Recommendation: Set X-Frame-Options: DENY or use CSP frame-ancestors.

Solution
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. 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.
References

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Frame-Options

Details	
Alert ID	10020-1
Alert Type	Passive
Status	release
Risk	Medium
CWE	1021
WASC	15
Technologies Targeted	All
Tags	CWE-1021 OWASP_2017_A06 OWASP_2021_A05 POLICY_PENTEST POLICY_QA_STD SYSTEMIC WSTG-V42-CLNT-09
More Info	Scan Rule Help

4.Subresource Integrity (SRI) Attribute Missing

Summary
The integrity attribute is missing on a script or link tag served by an external server. The integrity tag prevents an attacker who have gained access to this server from injecting a malicious content.

- **Impact:** Compromised third-party scripts
- **Recommendation:** Add `integrity` and `crossorigin` attributes to external resources

Solution
Provide a valid integrity attribute to the tag.
Other Info
The following hash was calculated (using base64 encoding of the output of the hash algorithm: SHA-384) for the script in question sha384-PJww2fZl501RXIQpYNSkUcg6ASX9Pec5LXs3IxrxDHLqWK7fzfiaV2W/kCr5Ps8G
References
- https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity

Details	
Alert ID	90003
Alert Type	Passive
Status	release
Risk	Medium
CWE	345
WASC	15
Technologies Targeted	All
Tags	CWE-345 OWASP_2017_A06 OWASP_2021_A05 POLICY_DEV_STD POLICY_PENTEST POLICY_QA_STD SYSTEMIC
More Info	Scan Rule Help

Low Risk Vulnerabilities

5.Cookie Without HttpOnly Flag

Summary
A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.

Impact: Increased CSRF risk
Recommendation: Use SameSite=Lax or SameSite=Strict.
Solution
Ensure that the HttpOnly flag is set for all cookies.
References

- https://owasp.org/www-community/HttpOnly

Details	
Alert ID	10010
Alert Type	Passive
Status	release
Risk	Low
CWE	1004
WASC	13
Technologies Targeted	All
Tags	CWE-1004 OWASP_2017_A06 OWASP_2021_A05 POLICY_DEV_STD POLICY_PENTEST POLICY_QA_STD SYSTEMIC WSTG-V42-SESS-02
More Info	Scan Rule Help

6.Cookie Without SameSite Attribute

Summary
A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.
Impact: Increased CSRF risk
Recommendation: Use SameSite=Lax or SameSite=Strict

Solution
Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.
Other Info
References

- https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-cookie-same-site

Details	
Alert ID	10054-1
Alert Type	Passive
Status	release
Risk	Low
CWE	1275
WASC	13
Technologies Targeted	All
Tags	CWE-1275 OWASP_2017_A05 OWASP_2021_A01 POLICY_DEV_STD POLICY_PENTEST POLICY_QA_STD SYSTEMIC WSTG-V42-SESS-02
More Info	Scan Rule Help

7.Cross-Domain JavaScript Source Inclusion

Summary
The page includes one or more script files from a third-party domain.

Impact: Trust on external domains
Recommendation: Host scripts locally or verify integrity.

Solution
Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.

Details	
Alert ID	**10017**
Alert Type	**Passive**
Status	**release**
Risk	**Low**
CWE	829
WASC	**15**
Technologies Targeted	**All**
Tags	CWE-829 OWASP_2021_A08 POLICY_DEV_STD POLICY_PENTEST POLICY_QA_STD SYSTEMIC
More Info	Scan Rule Help

8.Server Leaks Version Information

Summary

The web/application server is leaking the application it uses as a webserver via the “Server” HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to. This information alone, i.e. without a version string, is not very dangerous for the security of a server, nevertheless this information in the response header field is almost always useless and thus just an obsolete attacking vector.

Solution

Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Other Info

References

- https://httpd.apache.org/docs/current/mod/core.html#servertokens
- https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff648552(v=pandp.10)
- https://www.troyhunt.com/shhh-dont-let-your-response-headers/

Details	
Alert ID	10036-1
Alert Type	Passive
Status	release
Risk	Low
CWE	497
WASC	13
Technologies Targeted	All
Tags	CWE-497 OWASP_2017_A06 OWASP_2021_A05 POLICY_PENTEST POLICY_QA_STD SYSTEMIC WSTG-V42-INFO-02
More Info	Scan Rule Help

9.X-Content-Type-Options Header Missing

Summary
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
Impact: MIME-sniffing attacks
Recommendation: Set X-Content-Type-Options: nosniff.
Solution
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. 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.
Other Info
This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scan rule will not alert on client or server error responses.
References

- https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85)
- https://owasp.org/www-community/Security_Headers

Details	
Alert ID	10021
Alert Type	Passive
Status	release
Risk	Low
CWE	693
WASC	15
Technologies Targeted	All
Tags	CWE-693 OWASP_2017_A06 OWASP_2021_A05 POLICY_PENTEST POLICY_QA_STD SYSTEMIC
More Info	Scan Rule Help

Cross site scripting

Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

The vulnerability affects **http://testfire.net/index.jsp** , content

Discovered by Cross site scripting

PoC: URL encoded GET input content was set to inside_contact.htm<ScRiPt >ijUk(9037)</ScRiPt>)

The impact of this vulnerability

Malicious JavaScript has access to all the same objects as the rest of the web page, including access to cookies and local storage, which are often used to store session tokens. If an attacker can obtain a user's session cookie, they can then impersonate that user.