FUTURE INTERNS

Task 1: Vulnerability Assessment

Target : http://testphp.vulnweb.com

Prepared Abdirahman Hassan Mohamed

CIN: FIT/FEB26/CS6331

Organization : Future Interns

Executive Summary

I scanned testphp.vulnweb.com. I found “**Missing anti-clickjacking Header”, “Absence of Anti-CSRF tokens”, “X-Content-Type-Options Header missing”, “Strict-Transport-Security Header not set ”** vulnerabilities. The overall risk is Low and Medium.

The assessment revealed 4 primary security gaps related to missing HTTP security headers and session security. While no critical "High" risk vulnerabilities were exploited, the current configuration leaves users vulnerable to UI redressing and session hijacking.

1 . Missing Anti-Clickjacking Header (X-Frame-Options)

Risk Level: 🟡 Medium

Description: The response does not protect against ‘Clickjacking’ attacks. it should include either Content-security-policy with ‘frame-ancestors’ directive or X-Frame-Options.

Impact: An attacker could "overlay" this website on a malicious site, tricking users into clicking buttons they didn't intend to (Clickjacking).

Remediation: Add the X-Frame-Options: SAMEORIGIN or Content-Security-Policy: frame-ancestors 'self' header to the server response.

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,

2: Absence of Anti-CSRF Tokens

Risk Level: 🟡 Medium
Description: 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: A malicious site can force a user's browser to perform actions (like changing a password) on the target site without the user's knowledge.

Remediation: Implement unique CSRF tokens for every state-changing request (POST/PUT/DELETE).

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.

3: X-Content-Type-Options Header Missing

Risk Level: 🔵 Low

Description: 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.

Impact: Browsers might try to "guess" the content type of a file (MIME-sniffing), which could lead to the execution of malicious scripts disguised as images or text.

Remediation: Set the header X-Content-Type-Options: nosniff.
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.

4: Strict-Transport-Security (HSTS) Header Not Set

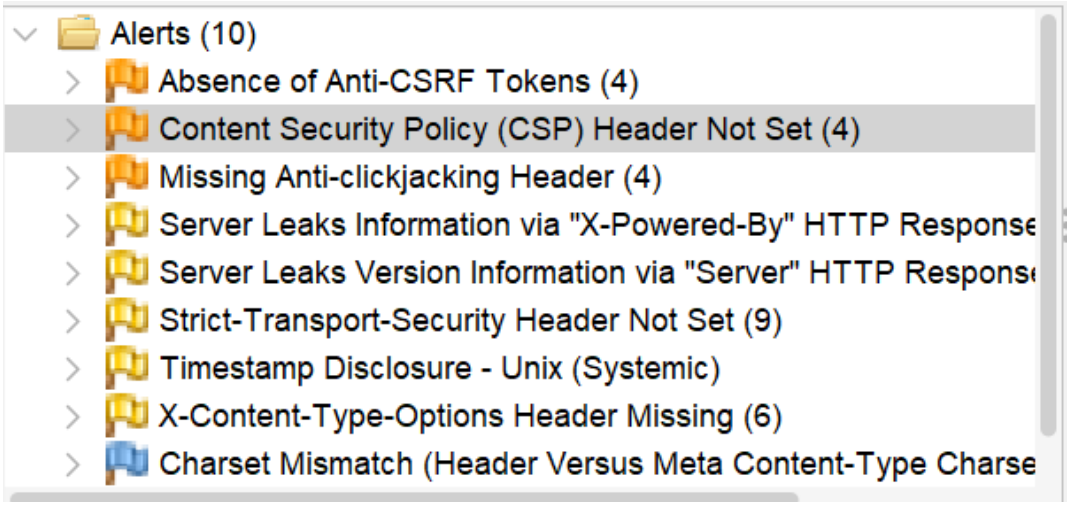
Risk Level: 🔵 Low

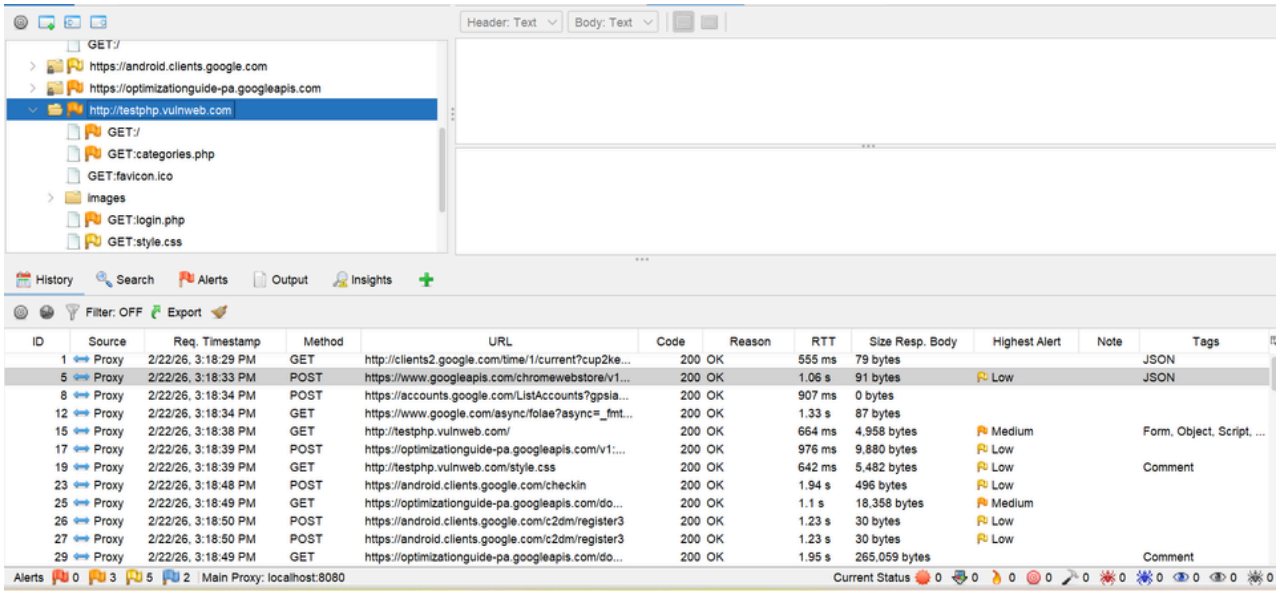
Description: HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTPS connections (i.e. HTTP layered over TLS/SSL). HSTS is an IETF standards track protocol and is specified in RFC 6797.

Impact: Users might be downgraded to an insecure HTTP connection, allowing "Man-in-the-Middle" (MitM) attacks to steal data.

Remediation: Enable the Strict-Transport-Security header with a proper max-age directive.
Ensure that your web server, application server, load balancer, etc. is configured to enforce Strict-Transport-Security.

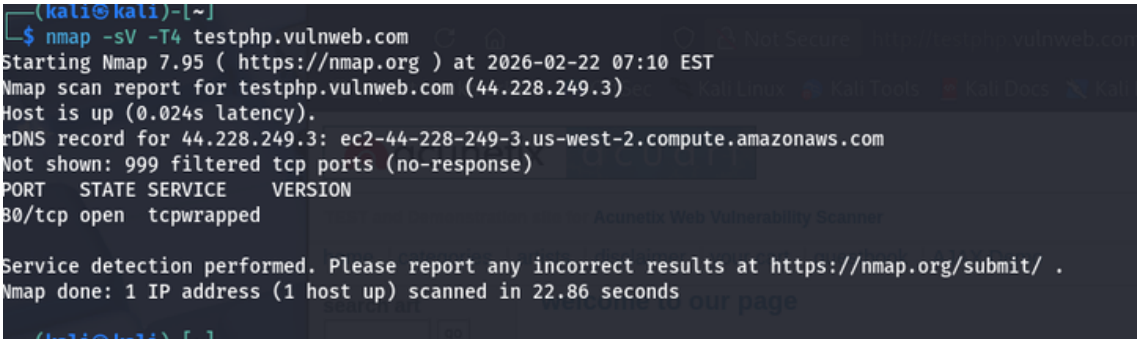
Evidence & Screenshots





Strict-Transport-Security Header Not Set

URL: https://www.googleapis.com/chromewebstore/v1.1/items/verify
Risk: Low
Confidence: High
Parameter:
Attack:
Evidence:
CWE ID: 319
WASC ID: 15
Source: Passive (10035 - Strict-Transport-Security Header)
Alert Reference: 10035-1
Input Vector:



Missing Anti-clickjacking Header

URL:	http://testphp.vulnweb.com/
Risk:	Medium
Confidence:	Medium
Parameter:	x-frame-options
Attack:	
Evidence:	
CWE ID:	1021
WASC ID:	15
Source:	Passive (10020 - Anti-clickjacking Header)
Alert Reference:	10020-1
Input Vector:	

X-Content-Type-Options Header Missing

URL:	http://testphp.vulnweb.com/style.css
Risk:	Low
Confidence:	Medium
Parameter:	x-content-type-options
Attack:	
Evidence:	
CWE ID:	693
WASC ID:	15
Source:	Passive (10021 - X-Content-Type-Options Header Missing)
Input Vector:	

Description:

Absence of Anti-CSRF Tokens

URL:	http://testphp.vulnweb.com/
Risk:	Medium
Confidence:	Low
Parameter:	
Attack:	
Evidence:	<form action="search.php?test=query" method="post">
CWE ID:	352
WASC ID:	9
Source:	Passive (10202 - Absence of Anti-CSRF Tokens)
Input Vector:	

Conclusion:

Final Note: Implementing these "Defense-in-Depth" measures will significantly harden the application against common automated attacks and UI-based exploits.