

Software and Cybersecurity Lab

CS445 Lab5

Name: Dipean Dasgupta

ID: 202151188

Task: Perform penetration testing on a selected website (public, non-government, or a specifically designed vulnerable website) and create a detailed penetration testing report based on your findings.

Target Website: www.owasp.org

Introduction:

This penetration test's main goal is to evaluate OWASP's IT infrastructure's security posture. This involves locating weak points that malevolent actors might use to jeopardize the confidentiality, availability, or integrity of vital systems. Finding internal and external threats is the goal, as is offering corrective measures to lessen hazards that are found.

The following important areas are covered by this penetration test:

- **Network infrastructure:** checking for incorrect settings, unpatched services, and open ports.
- **Web applications:** scanning for common vulnerabilities like Cross-Site Request Forgery (CSRF), SQL Injection (SQLi), and Cross-Site Scripting (XSS).
- **Servers and systems:** checking operating systems for configuration errors, out-of-date software, and inadequate user permissions.
- **Endpoints:** Workstations and mobile devices; these are the targets of potential malware or phishing attack entry points.

Methodology:

Reconnaissance / Information Gathering:

A. Nslookup

A Command-line tool used to query Domain Name System (DNS) and retrieve key details such as domain IP addresses, associated mail servers (MX records), and name servers (NS records).

```
C:\Users\Asus>nslookup www.owasp.org
Server:      gpon.net
Address:     192.168.1.1

Non-authoritative answer:
Name:        www.owasp.org
Addresses:   2606:4700:10::ac43:a27
             2606:4700:10::6816:1b4d
             2606:4700:10::6816:1a4d
             172.67.10.39
             104.22.26.77
             104.22.27.77
```

B. Whois:

A command-line tool used to gather registration and ownership information about a domain. By using **whois**, we were able to retrieve details like the domain registrant's name, contact information, and technical support contacts. These helps in understanding the potential entry points.

```
Domain Name: owasp.org
Registry Domain ID: 434f4e6cf20248cdbf9cefe1b292d77b-LROR
Registrar WHOIS Server: http://whois.godaddy.com
Registrar URL: http://www.whois.godaddy.com
Updated Date: 2024-07-07T13:31:38Z
Creation Date: 2001-09-21T17:00:36Z
Registry Expiry Date: 2031-09-21T17:00:36Z
Registrar: GoDaddy.com, LLC
Registrar IANA ID: 146
Registrar Abuse Contact Email: abuse@godaddy.com
Registrar Abuse Contact Phone: +1.4806242505
Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
Domain Status: clientRenewProhibited https://icann.org/epp#clientRenewProhibited
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
Registry Registrant ID: REDACTED FOR PRIVACY
Registrant Name: REDACTED FOR PRIVACY
Registrant Organization: Domains By Proxy, LLC
Registrant Street: REDACTED FOR PRIVACY
Registrant City: REDACTED FOR PRIVACY
Registrant State/Province: Arizona
Registrant Postal Code: REDACTED FOR PRIVACY
Registrant Country: US
Registrant Phone: REDACTED FOR PRIVACY
Registrant Phone Ext: REDACTED FOR PRIVACY
Registrant Fax: REDACTED FOR PRIVACY
Registrant Fax Ext: REDACTED FOR PRIVACY
Name Server: fay.ns.cloudflare.com
Name Server: west.ns.cloudflare.com
DNSSEC: unsigned
URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of WHOIS database: 2024-09-26T18:45:16Z <<<
```

Scanning

Tool: Nmap:

A popular open-source tool for network discovery and security audits is Nmap (Network Mapper). We actively scanned the target's network using nmap to find open ports, services that were using those ports, and information about the operating system. Among the key scanning methods used were:

Ping Scan, SYN Scan, Service scan, OS detection, Open Ports, traceroute

Scanning Initiation:

Nmap Output	Ports / Hosts	Topology	Host Details	Scans
nmap -T4 -A -v www.owasp.org				
Starting Nmap 7.95 (https://nmap.org) at 2024-09-27 16:29 India Standard Time				
NSE: Loaded 157 scripts for scanning.				
NSE: Script Pre-scanning.				
Initiating NSE at 16:29				
Completed NSE at 16:29, 0.00s elapsed				

Ping Scan:

Initiating Ping Scan at 16:29
Scanning www.owasp.org (172.67.10.39) [4 ports]
Completed Ping Scan at 16:29, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 16:29
Completed Parallel DNS resolution of 1 host. at 16:29, 0.03s elapsed





SYN Scan:

Initiating SYN Stealth Scan at 16:29
Scanning www.owasp.org (172.67.10.39) [1000 ports]
Discovered open port 443/tcp on 172.67.10.39
Discovered open port 8080/tcp on 172.67.10.39
Discovered open port 80/tcp on 172.67.10.39
Discovered open port 8443/tcp on 172.67.10.39
Completed SYN Stealth Scan at 16:29, 5.22s elapsed (1000 total ports)

Service scan

Initiating Service scan at 16:29
Scanning 4 services on www.owasp.org (172.67.10.39)
Completed Service scan at 16:30, 12.16s elapsed (4 services on 1 host)
Initiating OS detection (try #1) against www.owasp.org (172.67.10.39)
Retrying OS detection (try #2) against www.owasp.org (172.67.10.39)
Initiating Traceroute at 16:30
Completed Traceroute at 16:30, 3.03s elapsed
Initiating Parallel DNS resolution of 3 hosts. at 16:30
Completed Parallel DNS resolution of 3 hosts. at 16:30, 13.03s elapsed

Ports Found:

Nmap Output	Ports / Hosts	Topology	Host Details	Scans
Hostname	Port	Protocol	State	Version
 www.owasp.org (172.67.10.39)	80	tcp	open	Cloudflare http proxy
 www.owasp.org (172.67.10.39)	443	tcp	open	Cloudflare http proxy
 www.owasp.org (172.67.10.39)	8080	tcp	open	Cloudflare http proxy
 www.owasp.org (172.67.10.39)	8443	tcp	open	Cloudflare http proxy

Traceroute

Uptime guess: 0.001 days (since Fri Sep 27 16:30:09 2024)

Network Distance: 5 hops

TCP Sequence Prediction: Difficulty=260 (Good luck!)

IP ID Sequence Generation: All zeros

TRACEROUTE (using port 443/tcp)

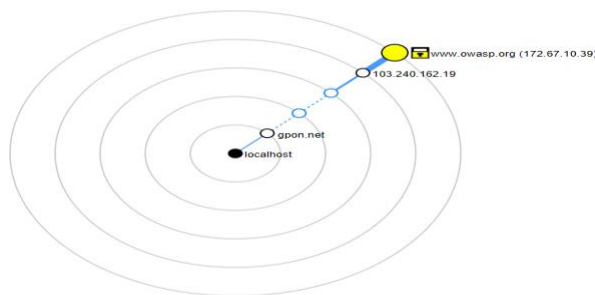
HOP RTT ADDRESS

1 2.00 ms gpon.net (192.168.1.1)

2 ... 3

4 5.00 ms 103.240.162.19

5 19.00 ms 172.67.10.39



Vulnerability Assessment and Exploitation:

Tool: OWASP ZAP (Zed Attack Proxy):

An open-source program called OWASP ZAP is used to test the security of online applications. We utilized it for both human testing of suspected vulnerabilities and automated vulnerability detection in web applications.



Automated Scan



This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically given permission to test.

URL to attack: Select...

Use traditional spider: ☒

Use ajax spider: If Modern with Chrome Headless

Attack Stop

Progress: Attack complete - see the Alerts tab for details of any issues found

Following attacks were performed by the Zap platform in the target website:

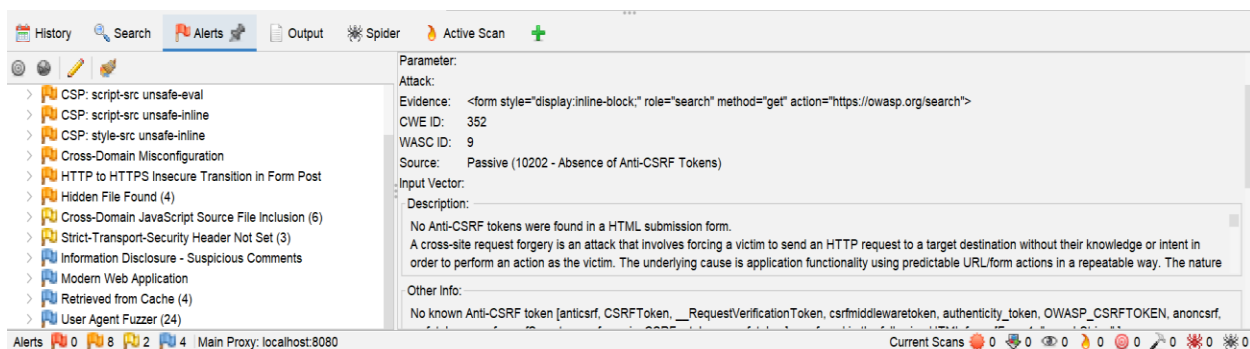
	Strength	Progress	Elapsed	Reqs	Alerts	Status
Analyser			00:00.727	1		
Plugin						
Path Traversal	Medium	100	00:00.009	0	0	Completed
Remote File Inclusion	Medium	100	00:00.003	0	0	Completed
Source Code Disclosure - /WEB-INF Folder	Medium	100	00:00.062	2	0	Completed
Heartbleed OpenSSL Vulnerability	Medium	100	00:00.177	3	0	Completed
Source Code Disclosure - CVE-2012-1823	Medium	100	00:00.177	3	0	Completed

Remote Code Execution - CVE-2012-1823	Medium	100	00:00.432	0	0	Completed
External Redirect	Medium	100	00:00.004	0	0	Completed
Server Side Include	Medium	100	00:00.003	0	0	Completed
Cross Site Scripting (Reflected)	Medium	100	00:00.004	0	0	Completed
Cross Site Scripting (Persistent) - Prime	Medium	100	00:00.003	0	0	Completed
Cross Site Scripting (Persistent) - Spider	Medium	100	00:00.054	3	0	Completed
Cross Site Scripting (Persistent)	Medium	100	00:00.002	0	0	Completed
SQL Injection	Medium	100	00:00.006	0	0	Completed
SQL Injection - MySQL	Medium	100	00:00.003	0	0	Completed
SQL Injection - Hypersonic SQL	Medium	100	00:00.003	0	0	Completed
SQL Injection - Oracle	Medium	100	00:00.002	0	0	Completed
SQL Injection - PostgreSQL	Medium	100	00:00.003	0	0	Completed
SQL Injection - SQLite	Medium	100	00:00.003	0	0	Completed
Cross Site Scripting (DOM Based)	Medium	100	02:13.899	370	0	Completed
SQL Injection - MsSQL	Medium	100	00:00.003	0	0	Completed
Log4Shell	Medium	100	00:00.001	4	0	Completed
Spring4Shell	Medium	100	00:00.512	7	0	Completed
Server Side Code Injection	Medium	100	00:00.003	0	0	Completed
Remote OS Command Injection	Medium	100	00:00.002	0	0	Completed
XPath Injection	Medium	100	00:00.001	0	0	Completed
XML External Entity Attack	Medium	100	00:00.002	0	0	Completed
Generic Padding Oracle	Medium	100	00:00.001	0	0	Completed
Cloud Metadata Potentially Exposed	Medium	100	00:39.032	4	0	Completed
Server Side Template Injection	Medium	100	00:39.036	0	0	Completed
Server Side Template Injection (Blind)	Medium	100	00:00.003	0	0	Completed
Directory Browsing	Medium	100	00:00.719	3	0	Completed
Buffer Overflow	Medium	100	00:00.002	0	0	Completed
Format String Error	Medium	100	00:00.002	0	0	Completed
CRLF Injection	Medium	100	00:00.002	0	0	Completed
Parameter Tampering	Medium	100	00:00.002	0	0	Completed
ELMAH Information Leak	Medium	100	00:00.039	1	0	Completed
Trace.axd Information Leak	Medium	100	00:00.040	2	0	Completed
.htaccess Information Leak	Medium	100	00:00.031	1	0	Completed
.env Information Leak	Medium	100	00:00.027	1	0	Completed
Spring Actuator Information Leak	Medium	100	00:01.414	2	0	Completed
Hidden File Finder	Medium	100	00:01.434	50	4	Completed
XSLT Injection	Medium	100	00:01.424	0	0	Completed
GET for POST	Medium	100	00:00.002	0	0	Completed

User Agent Fuzzer	Medium	100	00:06.808	36	24	Completed
Script Active Scan Rules	Medium	100	00:00.002	0	0	Completed
SOAP Action Spoofing	Medium	100	00:00.002	0	0	Completed
SOAP XML Injection	Medium	100	00:00.002	0	0	Completed
Totals			03:04.819	489	28	

TEST FINDINGS:

A total of 14 vulnerabilities and risks have been found from the tests conducted by ZAP on the targeted website. The image below shows all the 14 alerts in brief:



Of these 8 vulnerabilities are medium risk, 2 low risk and 4 on informational priority. Each vulnerability is presented in detail.

- 1. CSP: script-src unsafe-eval:** Allows JavaScript code execution using eval(), which can be exploited for injection attacks like XSS.
 Input Vector: Scripting (XSS), and data injection attacks.
 Risk: Medium
 Confidence: High
 Parameter: Content-Security-Policy
 Source: Passive (10055 - CSP)
 Reference:
<https://www.w3.org/TR/CSP/>
<https://caniuse.com/#search=content+security+policy>
<https://content-security-policy.com/>
<https://github.com/HtmlUnit/htmlunit-csp>
https://developers.google.com/web/fundamentals/security/csp#policy_applies_to_a_wide_variety_of_resources
- 2. Absence of Anti-CSRF Tokens:** vulnerable to CSRF attacks, allowing attackers to forge requests on behalf of authenticated users.
 Risk: Medium
 Confidence: Low
 Source: Passive(10202)
 Reference:

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

3. CSP: Wildcard Directive

The use of wildcards (*) in CSP directives permits content from any source, increasing the risk of malicious content being injected and executed on the site.

Risk: Medium

Confidence: High

Parameter: Content-Security-Policy

Source: Passive (10055 - CSP)

Reference:

<https://www.w3.org/TR/CSP/>

<https://caniuse.com/#search=content+security+policy>

<https://content-security-policy.com/>

4. CSP: script-src unsafe-inline

Permits inline JavaScript execution, increasing the risk of Cross-Site Scripting (XSS) attacks.

Risk: Medium

Confidence: High

Parameter: Content-Security-Policy

Source: Passive (10055 - CSP)

5. CSP: style-src unsafe-inline

Allows inline styles, which could be exploited to inject malicious code via style attributes.

Risk: Medium

Confidence: High

Parameter: Content-Security-Policy

Source: Passive (10055 - CSP)

6. Cross-Domain Misconfiguration

Incorrect configuration allows unauthorized access or manipulation of resources across different domains.

Risk: Medium

Confidence: High

Parameter: Access-Control-Allow-Origin

Source: Passive (10098 - Cross-Domain Misconfiguration)

Reference:

https://vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy

7. HTTP to HTTPS Insecure Transition in Form Post

Forms transition from HTTPS to HTTP, exposing sensitive data like credentials to attackers.

Risk: Medium

Confidence: Medium

Source: Passive (10041 - HTTP to HTTPS Insecure Transition in Form Post)

8. Hidden File Found:

Files that are not meant to be publicly accessible were discovered, potentially exposing sensitive information.

Risk: Medium

Confidence: Low

Source: Active (40035 - Hidden File Finder)

Reference:

<https://blog.hboeck.de/archives/892-Introducing-Snallygaster-a-Tool-to-Scan-for-Secrets-on-Web-Servers.html>

9. Cross-Domain JavaScript Source File Inclusion

External JavaScript files from different domains are included, which may introduce security risks from untrusted sources.

Risk: Low

Confidence: Medium

Source: Passive (10017 - Cross-Domain JavaScript Source File Inclusion)

10. Strict-Transport-Security Header Not Set

The absence of HSTS headers means the website is vulnerable to man-in-the-middle attacks due to lack of enforced HTTPS.

Risk: Low

Confidence: High

Reference:

https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

<https://owasp.org/www-community/Security-Headers>

https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security

Other Vulnerabilities:

Information Disclosure - Suspicious Comments: It was discovered that certain code comments contained sensitive information or debugging insights that may have helped attackers.

Modern Web Application: ZAP noted the usage of contemporary web frameworks, which may need for certain security setups.

Retrieved from Cache: Sensitive information that might be accessed by unauthorized persons may be exposed by cached answers.

User Agent Fuzzer : Based on the server's reaction to manipulated user agents, automated fuzzing of user-agent headers discovered a number of possible misconfigurations or vulnerabilities.

Recommendations and Solutions:

Recommendations/ solutions for each of the vulnerabilities listed by ZAP are given below:

- **Absence of Anti-CSRF Tokens:**

A vetted library or framework that does not allow this weakness to occur has to be used.

It will provide constructs that will make this weakness easier to avoid.

For example, using anti-CSRF packages such as the OWASP CSRFGuard.

- **CSP wildcard directive; CSP: script-src unsafe-eval, CSP: script-src unsafe-inline, CSP: style-src unsafe-inline:**

It has to be ensured that the Content-Security-Policy header is set correctly on your web server, application server, load balancer, etc.

- **Cross-Domain Misconfiguration:**

Ensuring no unauthenticated access to sensitive data exists (for example, by employing IP address white-listing). Also enable the web browser to enforce the Same Origin Policy (SOP) more strictly, configure the "Access-Control-Allow-Origin" HTTP header to a more restricted set of domains, or delete all CORS headers altogether.

- **HTTP to HTTPS Insecure Transition in Form Post:**

Using HTTPS for landing pages that host secure forms.

- **Hidden File Found:**

Properly examining whether the component is genuinely needed for production; if not, disable it. If so, make sure that access to it is subject to the proper authorization and authentication processes, or restrict exposure to internal systems, particular source IPs.

- **Cross-Domain JavaScript Source File Inclusion:**

Ensure the application's end users cannot control the sources from which JavaScript source files are loaded. Only source files from reliable sources should be loaded.

- **Strict-Transport-Security Header Not Set:**

Verification of Strict-Transport-Security has to be enforced by load balancer, web server, application server, etc.

- **Information Disclosure - Suspicious Comments**

Removal of any remarks that go back to material that could aid an attacker and addressal of any underlying issues they bring up.

- **Modern Web Application:**

Normal informational alert or warning so no such actions required.

- **Retrieved from Cache:**

Verification will be required that no private, sensitive, or user-specific information is contained in the response. If so, take into account using the following HTTP response headers to restrict or stop another user from storing and retrieving the material from the cache: Cache-Control: private, must-revalidate, no-store, and no-cache.

- **User Agent Fuzzer:**

Strict server-side validation should be used for user-agent strings to guarantee that only valid requests are handled. Furthermore, think about using more secure techniques for client identification and refrain from depending on the user-agent for essential functionality.

Conclusion:

Numerous critical and moderate vulnerabilities in the application were found throughout the penetration testing process of www.oawsp.org . These vulnerabilities included inappropriate handling of HTTP-to-HTTPS transitions, unsecured Content Security Policy setups, and the lack of anti-CSRF tokens. If these vulnerabilities are not fixed, they may result in many types of attacks, including data interception, Cross-Site Request Forgery (CSRF), and Cross-Site Scripting (XSS). But these problems can be efficiently resolved using the remediation suggestions offered, enhancing the system's overall security posture and guaranteeing improved defense against possible cyberattacks.

****ZAP FULL SCAN REPORT****

The full-fledged report can be accessed using the link below:

[ZAP Scanning Report 202151188](#)

-----END OF ASSIGNMENT-----