

ETHICAL HACKING INTERNSHIP

Name: Kalyani Gajanan Lonkar

Institutional Affiliation: Internship Studio

Email: kalyanilonkar5@gmail.com

Task : 2

**To test the website and find all possible vulnerabilities &
loopholes in it.**

<http://zero.webappsecurity.com/>

Using the NmapAutomator Vulnerability Scanner:

Firstly I used nikto on the given website, through the nikto scan.

I Got the Target IP 54.82.22.214

```
$ nikto -h http://zero.webappsecurity.com/
- Nikto v2.1.6
-----
+ Target IP:      54.82.22.214
+ Target Hostname: zero.webappsecurity.com
+ Target Port:    80
+ Start Time:     2022-08-24 15:28:24 (GMT5.5)
-----
```

Then using the **nmapAutomator** on the websites Ip 54.82.22.214

```
$ sudo ./nmapAutomator.sh -H 54.82.22.214 -t Vulns -output vulnerability
Running a Vulns scan on 54.82.22.214
No ping detected.. Will not use ping scans!
Host is likely running Unknown OS!
```

NmapAutomator did Port Scan

```
-----Starting Port Scan-----  
  
PORT      STATE SERVICE  
21/tcp    open  ftp  
80/tcp    open  http  
443/tcp   open  https  
554/tcp   open  rtsp  
1723/tcp  open  pptp  
8080/tcp  open  http-proxy  
  
-----Starting Vulns Scan-----
```

NampAutomator did Vulns Scan

Running Vuln Scan on Common Ports We got Vulnerabilities.

```
PORT      STATE SERVICE  VERSION  
21/tcp    open  ftp?  
80/tcp    open  http     Apache Tomcat/Coyote JSP engine 1.1  
| http-enum:  
| /admin/: Possible admin folder  
| /admin/index.html: Possible admin folder  
| /login.html: Possible admin folder  
| /manager/html/upload: Apache Tomcat (401 Unauthorized)  
| /manager/html: Apache Tomcat (401 Unauthorized)  
| /README.txt: Interesting, a readme.  
| /docs/: Potentially interesting folder  
|_ /errors/: Potentially interesting folder
```

1. Vulnerability

ssl-dh-params:

Transport Layer Security (TLS) Protocol DHE_EXPORT Ciphers Downgrade MitM (Logjam)

State : Vulnerable

IDs: CVE:CVE-2015-4000 BID:74733

The Transport Layer Security (TLS) protocol contains a flaw that is triggered when handling Diffie-Hellman key exchanges defined with the DHE_EXPORT cipher. This may allow a man-in-the-middle attacker to downgrade the security of a TLS session to 512-bit export-grade cryptography, which is significantly weaker, allowing the attacker to more easily break the encryption and monitor or tamper with the encrypted stream.

Disclosure date: 2015-5-19

```
ssl-dh-params:
VULNERABLE:
Transport Layer Security (TLS) Protocol DHE_EXPORT Ciphers Downgrade MitM (Logjam)
State: VULNERABLE
IDs: CVE:CVE-2015-4000 BID:74733
The Transport Layer Security (TLS) protocol contains a flaw that is
triggered when handling Diffie-Hellman key exchanges defined with
the DHE_EXPORT cipher. This may allow a man-in-the-middle attacker
to downgrade the security of a TLS session to 512-bit export-grade
cryptography, which is significantly weaker, allowing the attacker
to more easily break the encryption and monitor or tamper with
the encrypted stream.
Disclosure date: 2015-5-19
Check results:
EXPORT-GRADE DH GROUP 1
Cipher Suite: TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
Modulus Type: Safe prime
Modulus Source: mod_ssl 2.2.x/512-bit MODP group with safe prime modulus
Modulus Length: 512
Generator Length: 8
Public Key Length: 512
References:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-4000
https://weakdh.org
https://www.securityfocus.com/bid/74733
```

2. Vulnerability

ssl-ccs-injection:

SSL/TLS MITM vulnerability (CCS Injection)

State: Vulnerable

Risk factor: **High**

OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before 1.0.1h does not properly restrict processing of ChangeCipherSpec messages, which allows man-in-the-middle attackers to trigger use of a zero length master key in certain OpenSSL-to-OpenSSL communications, and consequently hijack sessions or obtain sensitive information, via crafted TLS handshake, aka the "CCS Injection" vulnerability.

```
ssl-ccs-injection:
VULNERABLE:
SSL/TLS MITM vulnerability (CCS Injection)
State: VULNERABLE
Risk factor: High
  OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before 1.0.1h
  does not properly restrict processing of ChangeCipherSpec messages,
  which allows man-in-the-middle attackers to trigger use of a zero
  length master key in certain OpenSSL-to-OpenSSL communications, and
  consequently hijack sessions or obtain sensitive information, via
  a crafted TLS handshake, aka the "CCS Injection" vulnerability.

References:
  https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0224
  http://www.cvedetails.com/cve/2014-0224
  http://www.openssl.org/news/secadv_20140605.txt
```

3. Vulnerability

http-vuln-cve2011-3192:

Apache byterange filter DoS

State: Vulnerable

IDs: CVE:CVE-2011-3192 BID:49303

The Apache web server is vulnerable to a denial of service attack when numerous overlapping byte ranges are requested.

Disclosure date: 2011-08-19

```
| http-vuln-cve2011-3192:
| VULNERABLE:
| Apache byterange filter DoS
| State: VULNERABLE
| IDs: CVE:CVE-2011-3192 BID:49303
| The Apache web server is vulnerable to a denial of service attack when numerous
| overlapping byte ranges are requested.
| Disclosure date: 2011-08-19
| References:
| https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3192
| https://www.tenable.com/plugins/nessus/55976
| https://seclists.org/fulldisclosure/2011/Aug/175
| https://www.securityfocus.com/bid/49303
|_
554/tcp open rtsp?
1723/tcp open pptp?
|_pptp-version: ERROR: Script execution failed (use -d to debug)
8080/tcp open http Apache Tomcat/Coyote JSP engine 1.1
|_http-dombased-xss: Couldn't find any DOM based XSS.
```

```

http-csrf:
Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=ec2-54-82-22-214.compute-1.amazonaws.com
Found the following possible CSRF vulnerabilities:

Path: http://ec2-54-82-22-214.compute-1.amazonaws.com:8080/
Form id: searchterm
Form action: /search.html

Path: http://ec2-54-82-22-214.compute-1.amazonaws.com:8080/index.html
Form id: searchterm
Form action: /search.html

http-enum:
/admin/: Possible admin folder
/admin/index.html: Possible admin folder
/login.html: Possible admin folder
/manager/html/upload: Apache Tomcat (401 Unauthorized)
/manager/html: Apache Tomcat (401 Unauthorized)
/README.txt: Interesting, a readme.
/docs/: Potentially interesting folder
/errors/: Potentially interesting folder
http-server-header: Apache-Coyote/1.1
http-stored-xss: Couldn't find any stored XSS vulnerabilities.

-----Finished all scans-----

```

4. Vulnerability

```

http-slowloris-check:
VULNERABLE:
Slowloris DOS attack
State: LIKELY VULNERABLE
IDs: CVE:CVE-2007-6750

Slowloris tries to keep many connections to the target web server open and hold
them open as long as possible. It accomplishes this by opening connections to
the target web server and sending a partial request. By doing so, it starves
the http server's resources causing Denial Of Service.

Disclosure date: 2009-09-17
References:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750
http://ha.ckers.org/slowloris/

http-csrf:
Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=ec2-54-82-22-214.compute-1.amazonaws.com
Found the following possible CSRF vulnerabilities:

Path: http://ec2-54-82-22-214.compute-1.amazonaws.com:8080/
Form id: searchterm
Form action: /search.html

Path: http://ec2-54-82-22-214.compute-1.amazonaws.com:8080/index.html
Form id: searchterm
Form action: /search.html

http-enum:
/admin/: Possible admin folder
/admin/index.html: Possible admin folder

```

Nmap Scan Report - Scanned at Tue Aug 23 18:52:27 2022

Scan Summary | zero.webappsecurity.com (54.82.22.214)

Scan Summary

Nmap 7.40 was initiated at Tue Aug 23 18:52:27 2022 with these arguments:
`nmap -v -oX=- --host-timeout=28800s -Pn -T4 -sT --webxml --max-retries=1 --open -p0-65355 zero.webappsecurity.com`

Verbosity: 1; Debug level 0

Nmap done at Tue Aug 23 18:54:04 2022; 1 IP address (1 host up) scanned in 96.18 seconds

54.82.22.214 / ec2-54-82-22-214.compute-1.amazonaws.com / zero.webappsecurity.com

Address

- 54.82.22.214 (ipv4)

Hostnames

- zero.webappsecurity.com (user)
- ec2-54-82-22-214.compute-1.amazonaws.com (PTR)

Ports

The 65353 ports scanned but not shown below are in state: **filtered**

- 65353 ports replied with: **no-responses**

Port	State (toggle closed [0] filtered [0])	Service	Reason	Product	Version	Extra info
80	tcp open	http	syn-ack			
443	tcp open	https	syn-ack			
8080	tcp open	http-proxy	syn-ack			

<p>High (CVSS: 7.5)</p> <p>NVT: Apache HTTP Server < 2.4.48 NULL Pointer Dereference Vulnerability - Windows</p>
<p>Product detection result</p> <p>cpe:/a:apache:http_server:2.2.6</p> <p>Detected by Apache HTTP Server Detection Consolidation (OID: 1.3.6.1.4.1.25623.1 ↪.0.117232)</p>
<p>Summary</p> <p>Apache HTTP Server is prone to a NULL pointer dereference vulnerability.</p>
<p>Vulnerability Detection Result</p> <p>Installed version: 2.2.6</p> <p>Fixed version: 2.4.48</p> <p>Installation path / port: 443/tcp</p>
<p>Impact</p> <p>Successful exploitation will allow an attacker to crash the server.</p>
<p>Solution:</p> <p>Solution type: VendorFix</p> <p>Update to version 2.4.48 or later.</p>
<p>Affected Software/OS</p> <p>Apache HTTP Server before version 2.4.48 on Windows.</p>
<p>Vulnerability Insight</p> <p>Apache HTTP Server protocol handler for the HTTP/2 protocol checks received request headers against the size limitations as configured for the server and used for the HTTP/1 protocol as well. On violation of these restrictions an HTTP response is sent to the client with a status code indicating why the request was rejected.</p> <p>This rejection response was not fully initialised in the HTTP/2 protocol handler if the offending header was the very first one received or appeared in a footer. This led to a NULL pointer dereference on initialised memory, crashing reliably the child process.</p>
<p>Vulnerability Detection Method</p> <p>Checks if a vulnerable version is present on the target host.</p> <p>Details: Apache HTTP Server < 2.4.48 NULL Pointer Dereference Vulnerability - Windows</p> <p>OID:1.3.6.1.4.1.25623.1.0.112904</p> <p>Version used: 2021-08-24T09:01:06Z</p>
<p>Product Detection Result</p> <p>Product: cpe:/a:apache:http_server:2.2.6</p> <p>Method: Apache HTTP Server Detection Consolidation</p>

High (CVSS: 10.0) NVT: Apache HTTP Server End of Life (EOL) Detection (Windows)
Product detection result cpe:/a:apache:http_server:2.2.6 Detected by Apache HTTP Server Detection Consolidation (OID: 1.3.6.1.4.1.25623.1 ↔.0.117232)
... continues on next page ...

... continued from previous page ...
Summary The Apache HTTP Server version on the remote host has reached the End of Life (EOL) and should not be used anymore.
Vulnerability Detection Result The "Apache HTTP Server" version on the remote host has reached the end of life. CPE: cpe:/a:apache:http_server:2.2.6 Installed version: 2.2.6 Location/URL: 443/tcp EOL version: 2.2 EOL date: 2017-12-31
Impact An EOL version of the Apache HTTP Server is not receiving any security updates from the vendor. Unfixed security vulnerabilities might be leveraged by an attacker to compromise the security of this host.

Site: <http://zero.webappsecurity.com>

Generated on Tue, 23 Aug 2022 18:53:02

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	4
Low	1
Informational	0
False Positives:	0

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	6
Cross-Domain Misconfiguration	Medium	16
Missing Anti-clickjacking Header	Medium	6
Vulnerable JS Library	Medium	2
X-Content-Type-Options Header Missing	Low	10

Alert Detail

Medium	Absence of Anti-CSRF Tokens
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>CSRF attacks are effective in a number of situations, including:</p> <ul style="list-style-type: none"> * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site. <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>
URL	http://zero.webappsecurity.com
Method	GET
Parameter	

Attack	
Evidence	<form action="/search.html" class="navbar-search pull-right" style="padding-right: 20px">
URL	http://zero.webappsecurity.com/
Method	GET
Parameter	
Attack	
Evidence	<form action="/search.html" class="navbar-search pull-right" style="padding-right: 20px">
URL	http://zero.webappsecurity.com/index.html
Method	GET
Parameter	
Attack	
Evidence	<form action="/search.html" class="navbar-search pull-right" style="padding-right: 20px">
URL	http://zero.webappsecurity.com/login.html
Method	GET
Parameter	
Attack	
Evidence	<form id="login_form" action="/signin.html" method="post" class="form-horizontal">
URL	http://zero.webappsecurity.com/online-banking.html
Method	GET
Parameter	
Attack	
Evidence	<form action="/search.html" class="navbar-search pull-right" style="padding-right: 20px">
URL	http://zero.webappsecurity.com/search.html?searchTerm=ZAP
Method	GET
Parameter	
Attack	
Evidence	<form action="/search.html" class="navbar-search pull-right" style="padding-right: 20px">
Instances	6
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>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>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>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>

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	6
Cross-Domain Misconfiguration	Medium	16
Missing Anti-clickjacking Header	Medium	6
Vulnerable JS Library	Medium	2
X-Content-Type-Options Header Missing	Low	10