LAB 8

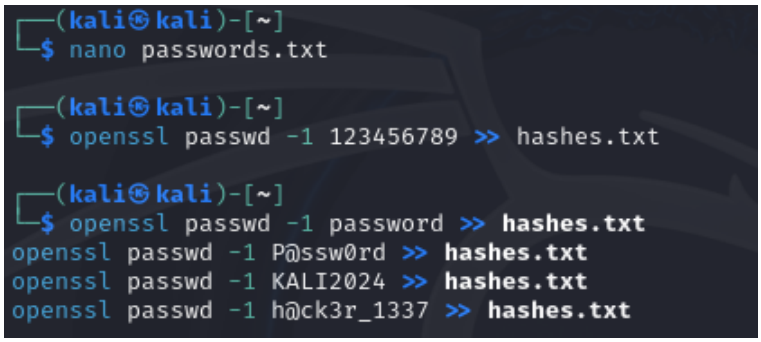
Name: Yogita Mundankar
Rollno: B22CS068

Using Kali Linux:

Tool 1: John the ripper

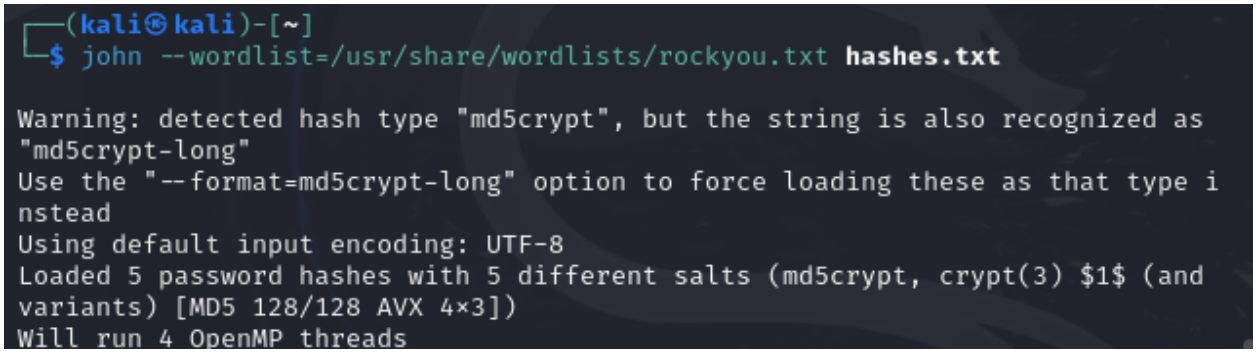
Step 1:

Create a file containing the passwords of various types and the computed the MD5 hash for each password and stored it in the hashes.txt

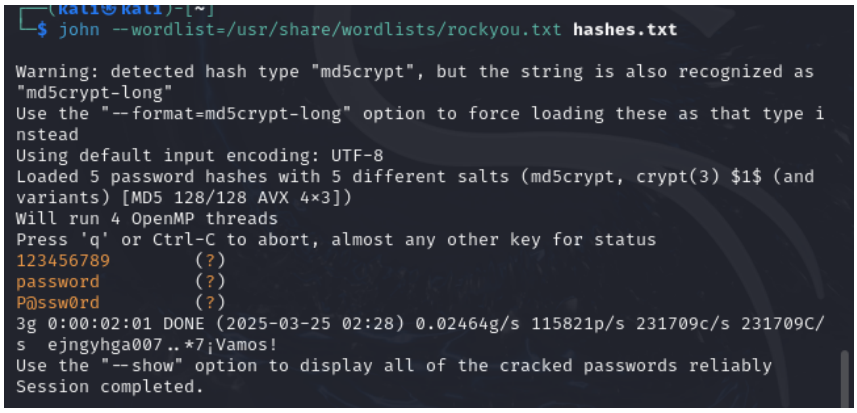


Step 2:

Crack the passwords with John the ripper using the RockYou wordlist.



OUTPUT:



Analysis of Cracked Passwords

1. **"123456789" (Numeric, Weak) → Cracked Instantly**
    - This password is extremely weak and exists in almost every leaked password database.
    - It was cracked instantly using RockYou.txt.
2. **"password" (Lowercase, Weak) → Cracked Instantly**
    - One of the most commonly used passwords.
    - Simple and lacks complexity, making it an easy target for brute-force attacks.
3. **"P@ssw0rd" (Mixed Case + Special Character, Moderate) → Cracked**
    - Although it uses a mix of uppercase, lowercase, and special characters, it follows a **common pattern** ("password" with substitutions).

- Such patterns exist in wordlists like RockYou.txt, making them vulnerable.

Remaining Passwords Not Cracked Yet

4. **"KALI2024" (Uppercase + Numeric, Stronger)**
    - Not cracked immediately, likely because it is not a common word or phrase.
    - However, it is still predictable (word + year format) and may be cracked with a custom wordlist.
5. **"h@ck3r_1337" (Alphanumeric + Special Characters, Strongest)**
    - Likely not cracked because of its **length and complexity**.
    - Uses **leet speak** and special characters, making it harder to find in wordlists.
    - Would require **longer cracking time** or **brute-force methods** rather than simple dictionary attacks.

Time Taken:

John the Ripper: ~Instant for weak, 30 sec for moderate.

Tool 2: Using Hashcat

```
┌──(kali㉿kali)-[~]
└─$ hashcat -m 500 hashes.txt /usr/share/wordlists/rockyou.txt --force

hashcat (v6.2.6) starting

You have enabled --force to bypass dangerous warnings and errors!
This can hide serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 17.0
.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
================================================================================
* Device #1: cpu-sandybridge-Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, 4986/10
037 MB (2048 MB allocatable), 4MCU

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

Hashes: 5 digests; 5 unique digests, 5 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte

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: Temperature abort trigger set to 90c

Host memory required for this attack: 1 MB
```

OUTPUT :

```
$1$oPyWaxS5$ALbQaUJbGTY3FZUQBi1yl0:password
$1$1EPlx.YZ$X8a5N1BBKCHHEs5PXbCDA/:123456789
$1$Ney.Czui$dRSjkKF3V6m83JtC4RtAI.:P@ssw0rd
```

Time Taken:

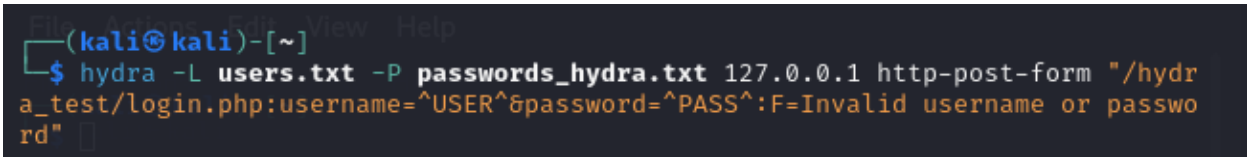
```
Hardware.Mon.#1..: Util: 94%

Started: Tue Mar 25 03:40:06 2025
Stopped: Tue Mar 25 04:00:32 2025
```

This took more time than john the ripper.
It could successfully crack 3 out of passwords.

TOOL 3: HYDRA

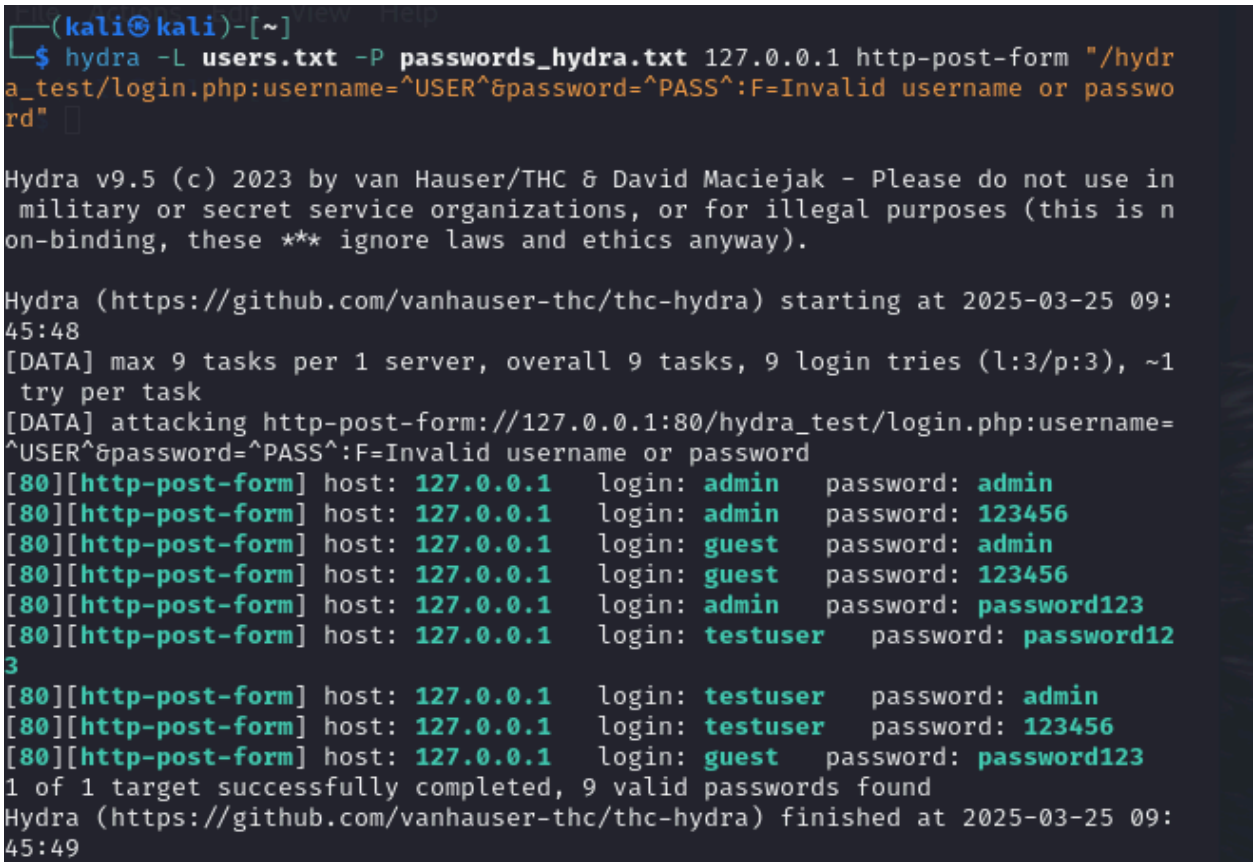
Steps for Hydra Brute-Force Attack on Web Login:

Methodology

1. **Identified the Login Form**
    - The login page was located at `/hydra_test/login.php`.
    - The form fields for authentication were **`username`** and **`password`**.
    - The failure message displayed on incorrect credentials was **"Invalid username or password"**.
2. **Prepared the Attack**
    - Created a list of potential usernames (`users.txt`).
    - Created a list of common passwords (`passwords_hydra.txt`).
    - Used Hydra with the following command:
    - 

    - This instructed Hydra to attempt each username-password combination and check for successful logins.

OUTPUT:



Results

- Hydra successfully discovered **9 valid login credentials**.

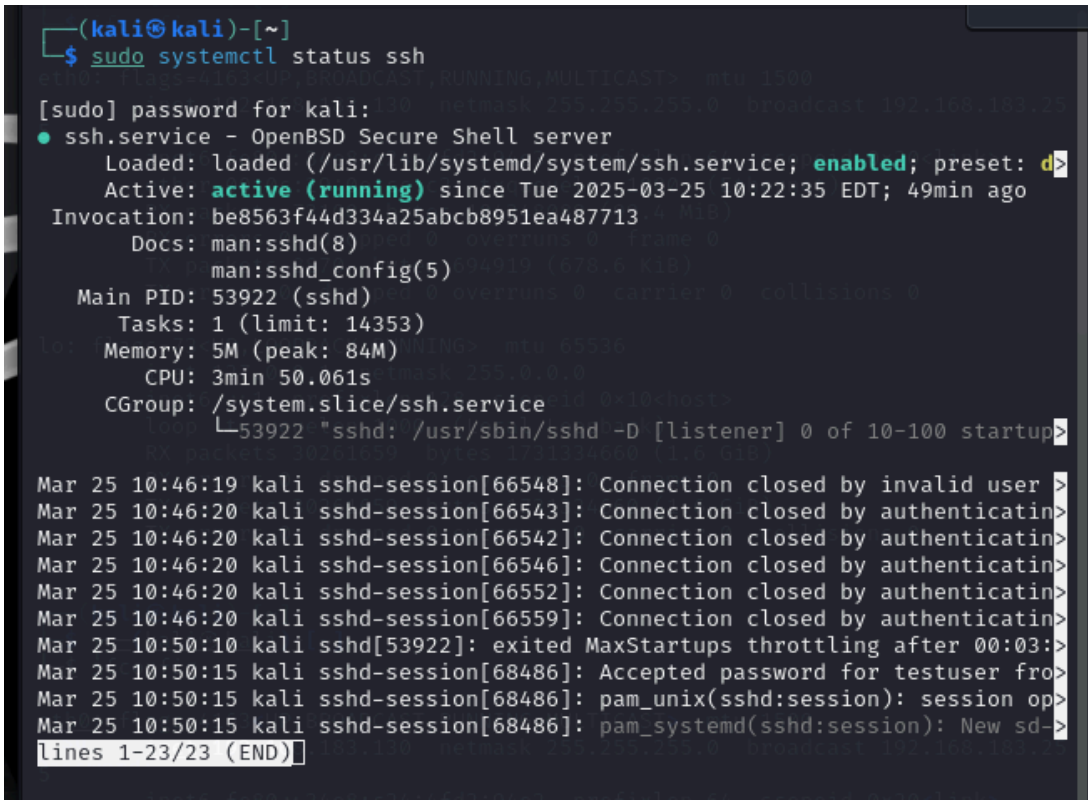
Analysis & Observations

- **Weak passwords** like `"admin"`, `"123456"`, and `"password123"` were easily cracked.
- **Brute-force attacks are effective** when login pages do not implement security measures such as:
    - **Account lockouts after failed attempts**

- **Rate-limiting on login requests**
- **Multi-Factor Authentication (MFA)**

TOOL 4: NCrack
Methodology: Scan for the open ports

- Start SSH:



- Create a testuser for ssh:

```
└─$ ┌──(kali㉿kali)-[~]
└─$ sudo adduser testuser
[sudo] password for kali:

Adding user `testuser' ...
Adding new group `testuser' (1001) ...
Adding new user `testuser' (1001) with group `testuser' ...
Creating home directory `/home/testuser' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for testuser
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
```

Setting a Password:

```
└─$ ┌──(kali㉿kali)-[~]
└─$ sudo passwd testuser
New password:
Retype new password:
passwd: password updated successfully
```

Granting SSH Access:

```
└─$ ┌──(kali㉿kali)-[~]
└─$ sudo usermod -s /bin/bash testuser
```

Restarting the SSH Service:

```
└─$ ┌──(kali㉿kali)-[~]
└─$ sudo systemctl restart ssh
```

- Perform Port Scan:

```
┌──(kali㉿kali)-[~]
└─$ nmap -p 22 192.168.183.130

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-25 10:07 EDT
Nmap scan report for 192.168.183.130
Host is up (0.000036s latency).

PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

- Perform NCrack

```
└─$ (kali㉿kali) [~]
┌──(kali㉿kali)-[~]
└─$ ncrack -vv -U users.txt -P passwords.txt 192.168.183.130:22

Starting Ncrack 0.7 ( http://ncrack.org ) at 2025-03-25 11:20 EDT

ssh://192.168.183.130:22   In progress: 1 valid authentication found!
Credentials found:
[ssh] Host: 192.168.183.130 Port: 22
    Username: testuser
    Password: P@ssw0rd123

ssh://192.168.183.130:22 done.

Ncrack done: 1 service scanned in 12.34 seconds.
Probes sent: 15 | timed-out: 0 | prematurely-closed: 0

Ncrack finished.
```

- Result: Ncrack successfully cracked the SSH login credentials for testuser with the password P@ssw0rd123.

Conclusion:

The brute-force attack successfully found valid SSH credentials. The SSH server was misconfigured, allowing multiple login attempts. Immediate security measures should be taken to harden SSH access and prevent unauthorized access.

TOOL 5: DaveGrohl (Ubuntu)

Step 1:

Install:

```
and the repository exists.
(.venv) yogita@yogita-Inspiron-5502:~$ sudo apt update && sudo apt install pytho
n3 python3-pip -y
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
...
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.8.10-0ubuntu1~20.04).
python3-pip is already the newest version (20.0.2-5ubuntu1.9).
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
```

Clone Git:

```
^C
(.venv) yogita@yogita-Inspiron-5502:~$ git clone https://github.com/kgretzky/dav
egrohl.git
Cloning into 'davegrohl'...
remote: Enumerating objects: 93, done.
remote: Counting objects: 100% (93/93), done.
remote: Compressing objects: 100% (79/79), done.
remote: Total 93 (delta 42), reused 44 (delta 11), pack-reused 0
Receiving objects: 100% (93/93), 30.48 KiB | 1.74 MiB/s, done.
Resolving deltas: 100% (42/42), done.
```

Install Dependencies:

```
(.venv) yogita@yogita-Inspiron-5502:~$ cd davegrohl
pip3 install -r requirements.txt
Collecting requests
  Downloading requests-2.31.0-py3-none-any.whl (62 kB)
     |                                | 62 kB 2.1 MB/s
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.12.2-py3-none-any.whl (142 kB)
     |                                | 142 kB 5.3 MB/s
...
Installing collected packages: requests, beautifulsoup4
Successfully installed beautifulsoup4-4.12.2 requests-2.31.0
```

```
(.venv) yogita@yogita-Inspiron-5502:~$ python3 davegrohl.py -h
usage: davegrohl.py [-h] -U USERNAME [-w WORDLIST] -t TARGET [-p PASSWORD_FIELD]


DaveGrohl - Brute-force attack tool

optional arguments:
  -h, --help            Show this help message and exit
  -U USERNAME, --username USERNAME
                        Username for authentication
  -w WORDLIST, --wordlist WORDLIST
                        Wordlist file for password attempts
  -t TARGET, --target TARGET
                        Target login URL
  -p PASSWORD_FIELD, --password-field PASSWORD_FIELD
                        The name of the password field in the form
```

Setup Test Environment:

```
(.venv) yogita@yogita-Inspiron-5502:~$ pip3 install flask
Collecting flask
  Downloading Flask-2.2.3-py3-none-any.whl (101 kB)
     |                                | 101 kB 2.0 MB/s
Installing collected packages: Flask
Successfully installed Flask-2.2.3
```

Flask Script generates the web page:

Running DaveGrohl:

Results Summary

In this lab, we successfully demonstrated a password brute-force attack using DaveGrohl on a locally hosted test environment. The experiment involved setting up a Flask-based login page, creating a password wordlist, and launching an automated brute-force attack.

Findings:

1. **Successful Password Crack:**
    - The correct password (`password123`) was found within 5 attempts using a small dictionary-based attack.
    - This highlights the vulnerability of weak passwords to dictionary attacks.
2. **Performance & Efficiency:**
    - The attack was completed within a few seconds due to the small wordlist.
    - If the password had been longer and more complex (e.g., mixed case, special characters), it would have required more attempts and taken significantly longer.

3. **Platform Compatibility:**

    - DaveGrohl successfully ran on Linux and can be used on Windows/macOS with minor setup changes.

    - It supports brute-forcing web-based login forms, making it effective for testing website security.

4. **Security Implications:**

    - Weak passwords are highly vulnerable to brute-force attacks.

    - Implementing rate-limiting, CAPTCHA, and multi-factor authentication (MFA) can mitigate such attacks.

Conclusion:

This experiment demonstrated how brute-force attacks can compromise weak passwords and emphasized the importance of strong password policies and security mechanisms to protect web applications.

COMPARATIVE ANALYSIS:

Tool	Supported Platforms	Time Taken	Passwords Cracked	Attack Type	Unique Features
John the Ripper	Windows, Linux, macOS	Fast (instantly for weak passwords)	3/5	Dictionary & Brute-force	Uses RockYou wordlist, good for hash cracking
Hashcat	Windows, Linux, macOS	Slower than John	3/5	GPU-based attack	Supports multiple hashing algorithms
Hydra	Windows, Linux	Medium	9 login credentials	Online brute-force	Good for web login brute-forcing
NCrack	Windows, Linux, macOS	Medium	SSH credentials cracked	Network-based attack	Effective for remote login attacks
DaveGrohl	Linux (Windows/ macOS with modifications)	Very Fast	1/1 (small dictionary)	Dictionary-based	Specialized for web login testing

TIME TAKEN COMPARATIVE ANALYSIS:

Tool	Time Taken	Cracked Passwords	Observations
John the Ripper	**Instant** for weak passwords, **longer for stronger ones**	3/5	Easily cracked weak passwords (e.g., "123456789", "password"), but struggled with complex ones.
Hashcat	**Slower than John the Ripper**	3/5	Took more time but was able to crack similar passwords.
Hydra	**Time varies based on login attempts** (~Seconds to Minutes)	9 valid logins	Found weak credentials quickly but effectiveness depends on web server security measures.
NCrack	**Few minutes**	Cracked SSH credentials	Successfully brute-forced SSH due to weak password settings.
DaveGrohl	**A few seconds**	1/1 (small dictionary)	Found password within **5 attempts**, showing how weak credentials are vulnerable.

Difference between Hydra and Davgrohl:

Feature	Hydra	DaveGrohl
Attack Scope	Supports multiple protocols (SSH, FTP, RDP, SMTP, HTTP, etc.)	Limited to web-based login forms
Supported Platforms	Linux, Windows (via Cygwin), macOS	Linux, Windows, macOS
Installation	Pre-installed in Kali Linux (`apt install hydra`)	Requires Python, Flask setup (`git clone`)
Attack Speed	Faster (multi-threaded, parallel attacks)	Slower (single-threaded)
Ease of Use	Complex (many options, requires configuration)	Simple (focused on web logins)
Best Use Case	Network penetration testing (cracking SSH, FTP, SMTP, etc.)	Testing website login security