

Lab 02 Report

Name: Madene Meriem Group: 01

October 16, 2025

Objective

Lab Title: Authentication in Practice

Main Goal:

Environment Setup

Virtual Machines:

VM Name	IP Address
Kali Linux	192.168.153.131
Metasploitable2	192.168.153.128

Tools Used:

- gpg encryption
- scp to transfer files securely
- nmap
- john the ripper password cracker
- hydra

Exercise 1: GPG Encryption and Decryption

Description:

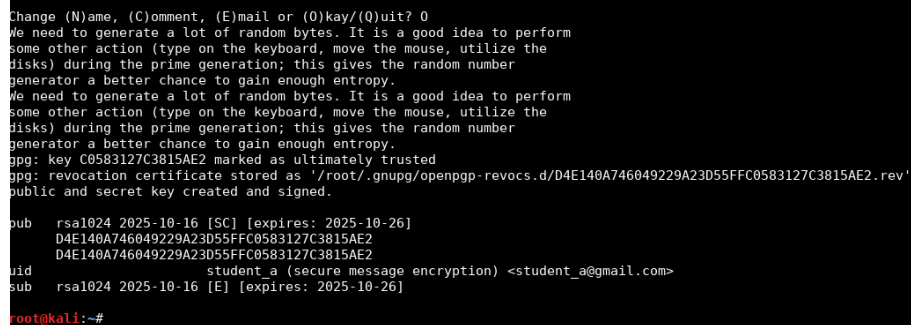
Encrypting a message and transfer it from metasploit machine to kali using GPG encryption and decryption .

Step 1: Generate GPG Keypair on Kali (Student A)

Commands Executed:

```
$ gpg --full-generate-key
```

Screenshot:



```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key C0583127C3815AE2 marked as ultimately trusted
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/D4E140A746049229A23D55FFC0583127C3815AE2.rev'
public and secret key created and signed.

pub   rsa1024 2025-10-16 [SC] [expires: 2025-10-26]
       D4E140A746049229A23D55FFC0583127C3815AE2
       D4E140A746049229A23D55FFC0583127C3815AE2
uid     student_a (secure message encryption) <student_a@gmail.com>
sub     rsa1024 2025-10-16 [E] [expires: 2025-10-26]

root@kali:~#
```

Analysis :

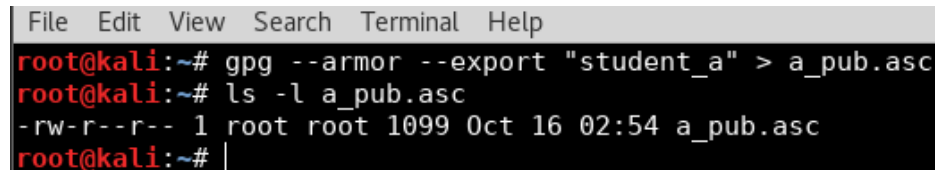
Generating a public key (RSA 1024bits expires in 10days) in order to use it in the encryption and the decryption process .

Step 2: Export Public Key

Commands Executed:

```
$ gpg --armor --export "student_a" > a_pub.asc
$ ls -l a_pub.asc
```

Screenshot:



```
File Edit View Search Terminal Help
root@kali:~# gpg --armor --export "student_a" > a_pub.asc
root@kali:~# ls -l a_pub.asc
-rw-r--r-- 1 root root 1099 Oct 16 02:54 a_pub.asc
root@kali:~# |
```

Analysis :

Exporting the public key of the "student_a" to create a shareable file that will be used in future decryption by the receiver .

Step 3: Transfer Public Key to Metasploitable2

Commands Executed:

```
$ scp a_pub.asc msfadmin@192.168.153.129:/home/msfadmin
```

Screenshot:

Analysis :

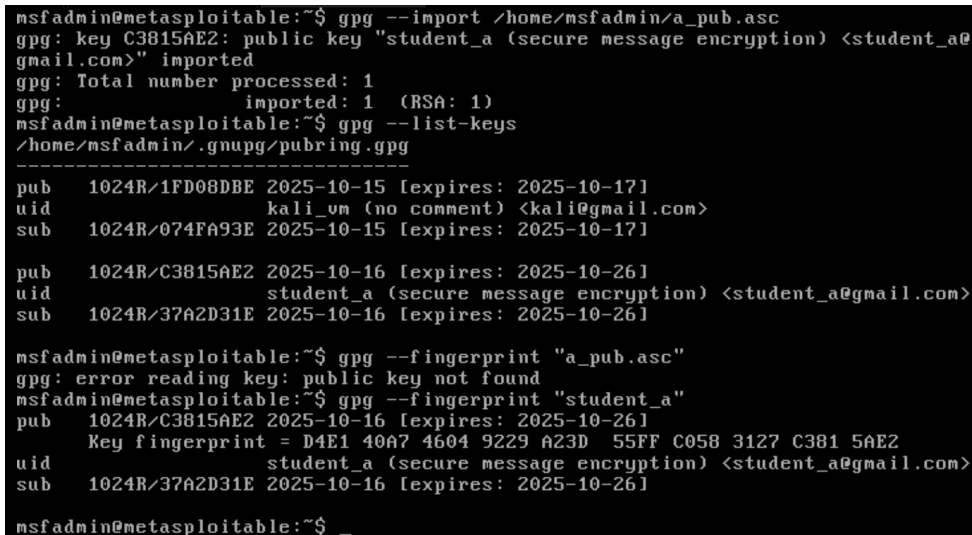
scp command help in copying the key file securely to the metasploit machine in order to use it in decrypting the message .

Step 4: Import and Verify Public Key on Metasploitable2

Commands Executed:

```
$ gpg --import /home/msfadmin/a_pub.asc
$ gpg --list-keys
$ gpg --fingerprint "a_pub.asc"
```

Screenshot:



```
msfadmin@metasploitable:~$ gpg --import /home/msfadmin/a_pub.asc
gpg: key C3815AE2: public key "student_a (secure message encryption) <student_a@gmail.com>" imported
gpg: Total number processed: 1
gpg:          imported: 1 (RSA: 1)
msfadmin@metasploitable:~$ gpg --list-keys
/home/msfadmin/.gnupg/pubring.gpg
-----
pub   1024R/1FD08DBE 2025-10-15 [expires: 2025-10-17]
uid           kali_vm (no comment) <kali@gmail.com>
sub   1024R/074FA93E 2025-10-15 [expires: 2025-10-17]

pub   1024R/C3815AE2 2025-10-16 [expires: 2025-10-26]
uid           student_a (secure message encryption) <student_a@gmail.com>
sub   1024R/37A2D31E 2025-10-16 [expires: 2025-10-26]

msfadmin@metasploitable:~$ gpg --fingerprint "a_pub.asc"
gpg: error reading key: public key not found
msfadmin@metasploitable:~$ gpg --fingerprint "student_a"
pub   1024R/C3815AE2 2025-10-16 [expires: 2025-10-26]
Key fingerprint = D4E1 40A7 4604 9229 A23D 55FF C058 3127 C381 5AE2
uid           student_a (secure message encryption) <student_a@gmail.com>
sub   1024R/37A2D31E 2025-10-16 [expires: 2025-10-26]

msfadmin@metasploitable:~$ _
```

Analysis :

Importing the key from the specified location to use it later Listing the keys to confirm the successful import GPG creates a shorter or summary hashed version of the key called fingerprint , its like an ID of the key's owner used to verify that the key is not changed by a mim or any other unauthorised entities .

DELIVERABLE - Public Key Fingerprint:

D4e1 40a7 4604 9229 a23d 55ff c058 3127 c381 5ae2

DELIVERABLE - Verification Method:

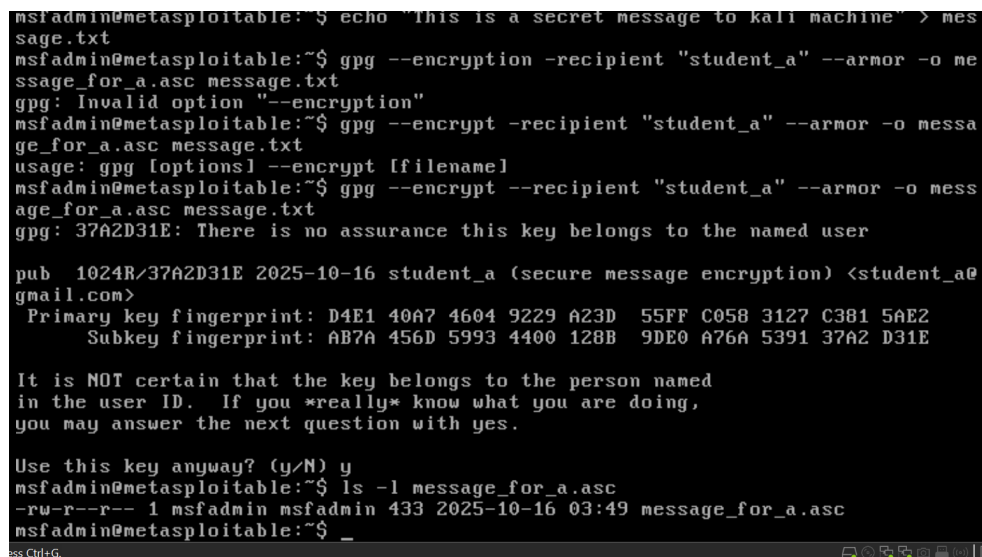
I verified the fingerprint by comparing it in both machines

Step 5: Create and Encrypt Message on Metasploitable2

Commands Executed:

```
$ echo "This is a secret message for kali machine" > message.txt
$ gpg --encrypt --recipient "student_a" --armor -o message_for_a.asc message.txt
$ ls -l message_for_a.asc
```

Screenshot:



```
msfadmin@metasploitable:~$ echo "This is a secret message for kali machine" > message.txt
msfadmin@metasploitable:~$ gpg --encrypt --recipient "student_a" --armor -o message_for_a.asc message.txt
gpg: Invalid option "--encryption"
msfadmin@metasploitable:~$ gpg --encrypt --recipient "student_a" --armor -o message_for_a.asc message.txt
usage: gpg [options] --encrypt [filename]
msfadmin@metasploitable:~$ gpg --encrypt --recipient "student_a" --armor -o message_for_a.asc message.txt
gpg: 37A2D31E: There is no assurance this key belongs to the named user

pub 1024R/37A2D31E 2025-10-16 student_a (secure message encryption) <student_a@gmail.com>
    Primary key fingerprint: D4E1 40A7 4604 9229 A23D 55FF C058 3127 C381 5AE2
    Subkey fingerprint: AB7A 456D 5993 4400 128B 9DE0 A76A 5391 37A2 D31E

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
msfadmin@metasploitable:~$ ls -l message_for_a.asc
-rw-r--r-- 1 msfadmin msfadmin 433 2025-10-16 03:49 message_for_a.asc
msfadmin@metasploitable:~$
```

DELIVERABLE - Encrypted File Content:



```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.6 (GNU/Linux)

hIwDp2pTkTei0x4BA/wMnfb0IB0webRlsGuf7Dach6bmFd9WPACUsxticWt+JMKF
Rx1BdS/QzODiUubt6nhlaUUL37NAM4PG4zDorDwoXXMfn+vvzG2ph7gR/3FUFdpQ
pAcVdQJQYb81AetnFM/zw5X1vP2UXvHRR47l2pdA9FAn+z8oq7RJewVR5MhqQtJo
Ae5J88P7wtCmQnwEs2mF+1E2/fU/ooQ0CoFHp+uuwpgyv1ubTQ8kvMB7Uv1Qnm7Q
UTl2Qka6lLvmcFfTgIfdG0jk05Cu9JvaJN6o+Uypl102hcGc2sU5+XkiNfviAYu1
5qupKscaiD0=
=6uHJ
-----END PGP MESSAGE-----
```

Step 6: Transfer Encrypted File to Kali

Commands Executed:

```
$ scp message_for_a.asc grp1@192.168.153.132:/home/grp1
```

Screenshot:

```

msfadmin@metasploitable:~$ ping 192.168.153.132
PING 192.168.153.132 (192.168.153.132) 56(84) bytes of data.
64 bytes from 192.168.153.132: icmp_seq=1 ttl=64 time=4.62 ms
64 bytes from 192.168.153.132: icmp_seq=2 ttl=64 time=1.94 ms
64 bytes from 192.168.153.132: icmp_seq=3 ttl=64 time=0.246 ms

--- 192.168.153.132 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 0.246/2.271/4.622/1.801 ms
msfadmin@metasploitable:~$ scp message_for_a.asc grp1@192.168.153.132:/home/grp1
ssh: connect to host 192.168.153.132 port 22: Connection refused
lost connection
msfadmin@metasploitable:~$

```

Analysis :

Successful secure message transfer from metasploit to kali machine with scp command .

Step 7: Decrypt Message on Kali

Commands Executed:

```
$ gpg --decrypt /home/grp1/message_for_a.asc > message.txt && cat message.txt
```

Screenshot:

```

root@kali:~# gpg --decrypt /home/grp1/message_for_kali.asc > message_decrypted.txt
gpg: encrypted with 1024-bit RSA key, ID A1185A1F074FA93E, created 2025-10-15
      "kali_vm (no comment) <kali@gmail.com>"
root@kali:~# cat message_decrypted.txt
This is a secret message from metasploit to kali.
root@kali:~#

```

Analysis:

Successful decryption process using the gpg keys in kali machine . Kali machine asked for the passphrase entered in the first step because the private key is protected .

Exercise 2: Reconnaissance & Discovery

Description:

Scanning metasploit machine to look for any authentication related services

Commands Executed:

```
$ nmap -sC -sV -p 1-10000 -o metasploit_scan 192.168.153.129
```

DELIVERABLE - Screenshot of Scan Results:

```

Starting Nmap 7.50 ( https://nmap.org ) at 2025-10-16 05:01 EDT
Nmap scan report for 192.168.153.129
Host is up (0.0025s latency).
Not shown: 9974 closed ports
PORT      STATE SERVICE        VERSION
21/tcp    open  ftp            vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp    open  ssh            OpenSSH 4.7p1 Debian Bubuuntu (protocol 2.0)
|_ssh-hostkey:
|_ 1924 60:08:f1:c1:c0:5f:6a:74:d6:9b:24:fa:c4:d5:6c:cd (DSA)
|_ 2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:bl:24:3d:e8:f3 (RSA)
23/tcp    open  telnet?
25/tcp    open  smtp?
|_setp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
|_ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside US/countryName=XX
|_Not valid before: 2010-03-17T14:07:45
|_Not valid after: 2010-04-16T14:07:45
|_ssl-date: 2025-10-16T08:22:18+00:00; -55m32s from scanner time.
33/tcp    open  domain         ISC BIND 9.4.2
|_dns-nsid:
|_bind.version: 9.4.2
80/tcp    open  http           Apache httpd 2.2.8 ((Ubuntu) DAV/2)
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_http-title: Metasploitable2 - Linux
111/tcp   open  rpcbind        2 (RPC #100000)
|_rpcinfo:
|_  program version port/proto service
|_  100000 2 111/tcp rpcbind
|_  100000 2 111/udp rpcbind
|_  100003 2,3,4 2049/tcp nfs
|_  100003 2,3,4 2049/udp nfs
|_  100005 1,2,3 38513/udp mountd
|_  100005 1,2,3 46379/tcp mountd
|_  100021 1,3,4 44523/udp nlockmgr
|_  100021 1,3,4 44827/tcp nlockmgr
|_  100024 1 43890/udp status
|_  100024 1 60036/tcp status
139/tcp   open  netbios-ssn    Samba smb2 3.0.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn    Samba smb2 3.0.20-Debian (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn    Samba smb2 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp   open  exec?
513/tcp   open  login?
514/tcp   open  shell?
1099/tcp  open  java-rmi        Java RMI Registry
1524/tcp  open  shell           Metasploitable root shell
2049/tcp  open  nfs             2-4 (RPC #100003)
2121/tcp  open  ccproxy-ftp?
3306/tcp  open  mysql?
|_mysql-info: ERROR: Script execution failed (use -d to debug)
3532/tcp  open  distccd         distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql      PostgreSQL DB 8.3.0 - 8.3.7
|_ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside US/countryName=XX
|_Not valid before: 2010-03-17T14:07:45
|_Not valid after: 2010-04-16T14:07:45
|_ssl-date: 2025-10-16T08:22:18+00:00; -55m32s from scanner time.
5900/tcp  open  vnc             VNC (protocol 3.3)
|_vnc-info:
|_  Protocol version: 3.3
|_  Security types:
|_  VNC Authentication (2)
5989/tcp  open  X11             (access denied)
6667/tcp  open  irc             UnrealIRCd
|_irc-info:
|_  users: 2.0
|_  servers: 1
|_  users: 2
|_  lservers: 0
|_  server: irc.Metasploitable.LAN
|_  version: Unreal3.2.8.1: irc.Metasploitable.LAN
|_  uptime: 0 days, 0:55:46
|_  source ident: nmap
|_  source host: 96215C01.576B88B8.FFFA6D49.IP
|_  error: Closing Link: faybelint[192.168.153.132] (Quit: faybelint)
6697/tcp  open  irc             UnrealIRCd
|_irc-info:
|_  users: 1.0
|_  servers: 1
|_  users: 1
|_  lservers: 0
|_  server: irc.Metasploitable.LAN
|_  version: Unreal3.2.8.1: irc.Metasploitable.LAN
|_  uptime: 0 days, 0:55:46
|_  source ident: nmap
|_  source host: 96215C01.576B88B8.FFFA6D49.IP
|_  error: Closing Link: idshfhut[192.168.153.132] (Quit: idshfhut)
8089/tcp  open  ajp13          Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTIONS request
8188/tcp  open  http           Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/5.5
8787/tcp  open  drb            Ruby DRB RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbb)
MAC Address: 08:0C:29:3E:23:CE (VMware)
Service Info: Hosts: localhost, irc.Metasploitable.LAN; OS: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: -55m32s, deviation: 0s, median: -55m32s
|_nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_ smb-os-discovery:
|_  OS: Unix (Samba 3.0.20-Debian)
|_  NetBIOS computer name:
|_  Workgroup: WORKGROUP\*80
|_  System time: 2025-10-16T04:22:04-04:00

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1048.08 seconds

```

Authentication Services Identified:

Service	Port
FTP	21
ssh	22

Analysis:

Nmap results show multiple open ports , one of them are related to authentication (FTP & SSH) , these security breaches can lead to multiples security issues like exploiting weakness , brute-forcing credentials , gaining shell access ..etc

Exercise 3: Simple Hashing & Cracking

Description:

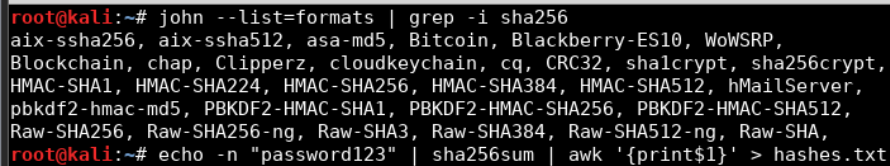
Password cracking using a word list with John the ripper (password cracking tool)

Task 1: Compute SHA-256 Hash

Commands Executed:

```
$ john --list=formats | grep -i sha256
$ echo -n "password123" | sha256sum | awk '{print$1}' > hashes.txt
```

Screenshot:



```
root@kali:~# john --list=formats | grep -i sha256
aix-ssha256, aix-ssha512, asa-md5, Bitcoin, Blackberry-ES10, WoWSRP,
Blockchain, chap, Clipperz, cloudkeychain, cq, CRC32, sha1crypt, sha256crypt,
HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, HMAC-SHA384, HMAC-SHA512, hMailServer,
pbkdf2-hmac-md5, PBKDF2-HMAC-SHA1, PBKDF2-HMAC-SHA256, PBKDF2-HMAC-SHA512,
Raw-SHA256, Raw-SHA256-ng, Raw-SHA3, Raw-SHA384, Raw-SHA512-ng, Raw-SHA,
root@kali:~# echo -n "password123" | sha256sum | awk '{print$1}' > hashes.txt
```

Key Findings:

- Chosen password: password123
- SHA-256 hash: ef92b778bafef771e89245b89ecbc08a44a4e166c06659911881f383d4473e94f (in base64)

Analysis:

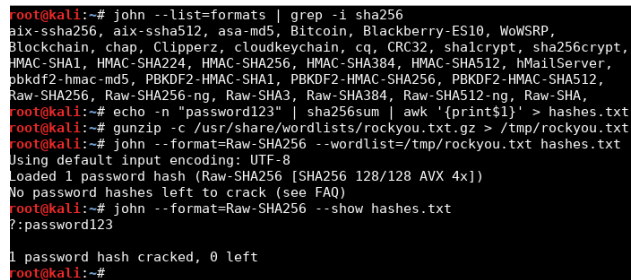
Hashing the password using Raw SHA256, this method is fast but easy to crack and vulnerable especially to wordlist attacks if the password is common.

Task 2: Crack the Hash

Commands Executed:

```
$ gunzip -c /usr/share/wordlists/rockyou.txt.gz > /tmp/rockyou.txt
$ john --format=Raw-SHA256 --wordlist=/tmp/rockyou.txt hashes.txt
$ john --format=Raw-SHA256 --show hashes.txt
```

Screenshot:



```
root@kali:~# john --list=formats | grep -i sha256
aix-ssha256, aix-ssha512, asa-md5, Bitcoin, Blackberry-ES10, WoWSRP,
Blockchain, chap, Clipperz, cloudkeychain, cq, CRC32, sha1crypt, sha256crypt,
HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, HMAC-SHA384, HMAC-SHA512, hMailServer,
pbkdf2-hmac-md5, PBKDF2-HMAC-SHA1, PBKDF2-HMAC-SHA256, PBKDF2-HMAC-SHA512,
Raw-SHA256, Raw-SHA256-ng, Raw-SHA3, Raw-SHA384, Raw-SHA512-ng, Raw-SHA,
root@kali:~# echo -n "password123" | sha256sum | awk '{print$1}' > hashes.txt
root@kali:~# gunzip -c /usr/share/wordlists/rockyou.txt.gz > /tmp/rockyou.txt
root@kali:~# john --format=Raw-SHA256 --wordlist=/tmp/rockyou.txt hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 128/128 AVX 4x])
No password hashes left to crack (see FAQ)
root@kali:~# john --format=Raw-SHA256 --show hashes.txt
?:password123

1 password hash cracked, 0 left
root@kali:~#
```

Cracking Result:

?:password123

1 password hash cracked, 0 left

DELIVERABLE - Question 1: Name two solutions to fix the flaw:

1. Using a strong hashing algorithm to ensure the security of the password
2. The use of strong password and non common ones that contains characters (small and caps), numbers, special characters .. etc so that the password cannot be cracked easily

DELIVERABLE - Question 2: Implementation of chosen solution:

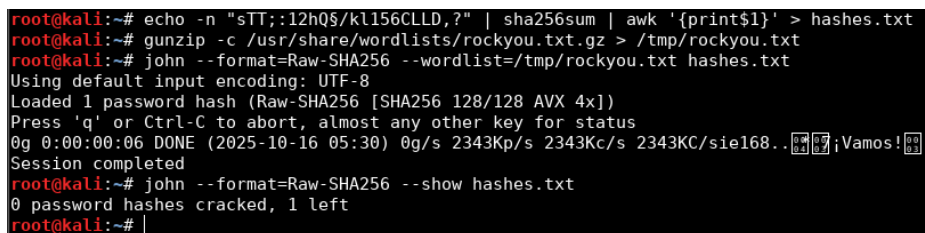
(Describe which solution you chose and how you implemented it)

Using a stronger password like : sTT;:12hQ\$/kl156CLLD,?

Commands for Implementation:

```
$ echo -n "sTT;:12hQ$/kl156CLLD,?" | sha256sum | awk '{print$1}' > hashes.txt
```

Screenshot of Implementation:



```
root@kali:~# echo -n "sTT;:12hQ$/kl156CLLD,?" | sha256sum | awk '{print$1}' > hashes.txt
root@kali:~# gunzip -c /usr/share/wordlists/rockyou.txt.gz > /tmp/rockyou.txt
root@kali:~# john --format=Raw-SHA256 --wordlist=/tmp/rockyou.txt hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 128/128 AVX 4x])
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:06 DONE (2025-10-16 05:30) 0g/s 2343Kp/s 2343Kc/s 2343KC/sie168..[?][?];Vamos![?][?]
Session completed
root@kali:~# john --format=Raw-SHA256 --show hashes.txt
0 password hashes cracked, 1 left
root@kali:~#
```

Analysis:

The use of strong password took a longer time to hash than the weak one . Also john the ripper failed to crack the password as it is not in the word list

Exercise 4: Simulated Online Attack & Defensive iptables Rule

Description:

Simulate a small brute-force using hydra tools and block the attacker's IP with iptables.

Step 1: Create Wordlist

Commands Executed:

```
$ cat > passlist.txt # followed by the list
```

Screenshot:


```

root@kali:~# cat passlist.txt
msfadmin
msf
msf4dm1n
password
password123
123456
12345678
1234
12345
123456789
admin
admin123
ftp
ftpuer
anonymous
guest
guest123
toor
root
root123
default
changeme
letmein
welcome
qwerty
qwerty123
pass
passwd
passwd
login
user
user123
test
test123
123qwe
abc123
iloveyou

```

Step 2: Launch Brute-Force Attack with Hydra

Commands Executed:

```
$ hydra -l msfadmin -P passlist.txt ftp://192.168.153.129 -t 4
```

Screenshot:

```

root@kali:~# hydra -l msfadmin -P passlist.txt ftp://192.168.153.129 -t 4
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2025-10-16 05:54:07
[DATA] max 4 tasks per 1 server, overall 64 tasks, 47 login tries (l:1/p:47), ~0 tries per task
[DATA] attacking service ftp on port 21
[21][ftp] host: 192.168.153.129  login: msfadmin  password: msfadmin
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2025-10-16 05:54:14
root@kali:~#

```

Attack Result:

Successful login to the FTP service using the hydra tool

Step 3: Block Kali IP with iptables on Metasploitable2

Commands Executed:

```
$ sudo iptables -A INPUT -s 192.168.153.132 -j DROP
```

Screenshot:

```

msfadmin@metasploitable:~$ sudo iptables -l
[sudo] password for msfadmin:
iptables v1.3.8: Unknown arg '-l'
Try 'iptables -h' or 'iptables --help' for more information.
msfadmin@metasploitable:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
msfadmin@metasploitable:~$ sudo iptable -A INPUT -s 192.168.153.132 -j DROP
sudo: iptable: command not found
msfadmin@metasploitable:~$ sudo iptables -A INPUT -s 192.168.153.132 -j DROP
msfadmin@metasploitable:~$

```

Blocking Rule Applied:

Applied rule is DROP which blocks all the incoming packets from the specified ip address (kali machine's address)

Step 4: Re-run Hydra After Blocking

Commands Executed:

```
$ hydra -l msfadmin -P passlist.txt ftp://192.168.153.129 -t 4
```

Screenshot:

```

root@kali:~# hydra -l msfadmin -P passlist.txt ftp://192.168.153.129 -t 4
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2025-10-16 07:02:00
[DATA] max 4 tasks per 1 server, overall 64 tasks, 47 login tries (l:1/p:47), ~0 tries per task
[DATA] attacking service ftp on port 21
[STATUS] 6.00 tries/min, 6 tries in 00:01h, 43 to do in 00:08h, 4 active
[STATUS] 7.00 tries/min, 14 tries in 00:02h, 43 to do in 00:07h, 4 active
[STATUS] 7.33 tries/min, 22 tries in 00:03h, 43 to do in 00:06h, 4 active
[STATUS] 7.50 tries/min, 30 tries in 00:04h, 43 to do in 00:06h, 4 active
[STATUS] 7.60 tries/min, 38 tries in 00:05h, 43 to do in 00:06h, 4 active
[STATUS] 7.67 tries/min, 46 tries in 00:06h, 43 to do in 00:06h, 4 active
[STATUS] 7.71 tries/min, 54 tries in 00:07h, 43 to do in 00:06h, 4 active
[STATUS] 7.25 tries/min, 58 tries in 00:08h, 43 to do in 00:06h, 4 active
[STATUS] 3.22 tries/min, 62 tries in 00:10h, 43 to do in 00:14h, 4 active
[STATUS] 3.46 tries/min, 70 tries in 00:20h, 43 to do in 00:13h, 4 active
[STATUS] 3.67 tries/min, 78 tries in 00:21h, 43 to do in 00:12h, 4 active
[STATUS] 3.78 tries/min, 84 tries in 00:22h, 43 to do in 00:12h, 4 active
[STATUS] 3.87 tries/min, 90 tries in 00:23h, 43 to do in 00:12h, 4 active
[STATUS] 4.04 tries/min, 98 tries in 00:24h, 43 to do in 00:11h, 4 active
*The session file ./hydra.restore was written. Type "hydra -R" to resume session.
root@kali:~#

```

Result After Blocking:

Failed login attempts because of the blocked ip address , meaning that you can't login to FTP service from that ip again.

Analysis:

Dropping the ip address with iptables in metasploits successfully blocked that ip from logging in remotely to the FTP service . But we still can log in from a different ip , so it is better practice to use a more secure service like FTPS .

Conclusion

Summary:

In this lab we performed many security basics and concepts :

- GPG encryption and how does it work and transfer messages between two users in a secure way
- Discovery and scanning metasploit machine to check any open ports especially authentication ports like FTP and SSH
- Simple password cracking and hashing using password cracking tools like John the ripper and SHA256 hashing algorithm
- Simulate a remote brute force attack on a target machine using hydra tool with a passlist over ftp service which was successful .

Security Implications:

- Weak passwords can be cracked easily in short time due to the lack of entropy
- Unencrypted authentication services leak credentials to unauthorized entities
- The lack of firewall rules can lead to unauthorized access and many other security threats

Lessons Learned:

- Always use strong passwords and non guessable ones , and use strong hashing algorithms
- Use encrypted services to prevent credentials leaks
- Use strong and secure rules in the Firewall to keep you system secure .