Lab15 Password Cracking Using John the Ripper and Hashcat
Author:Shashangka

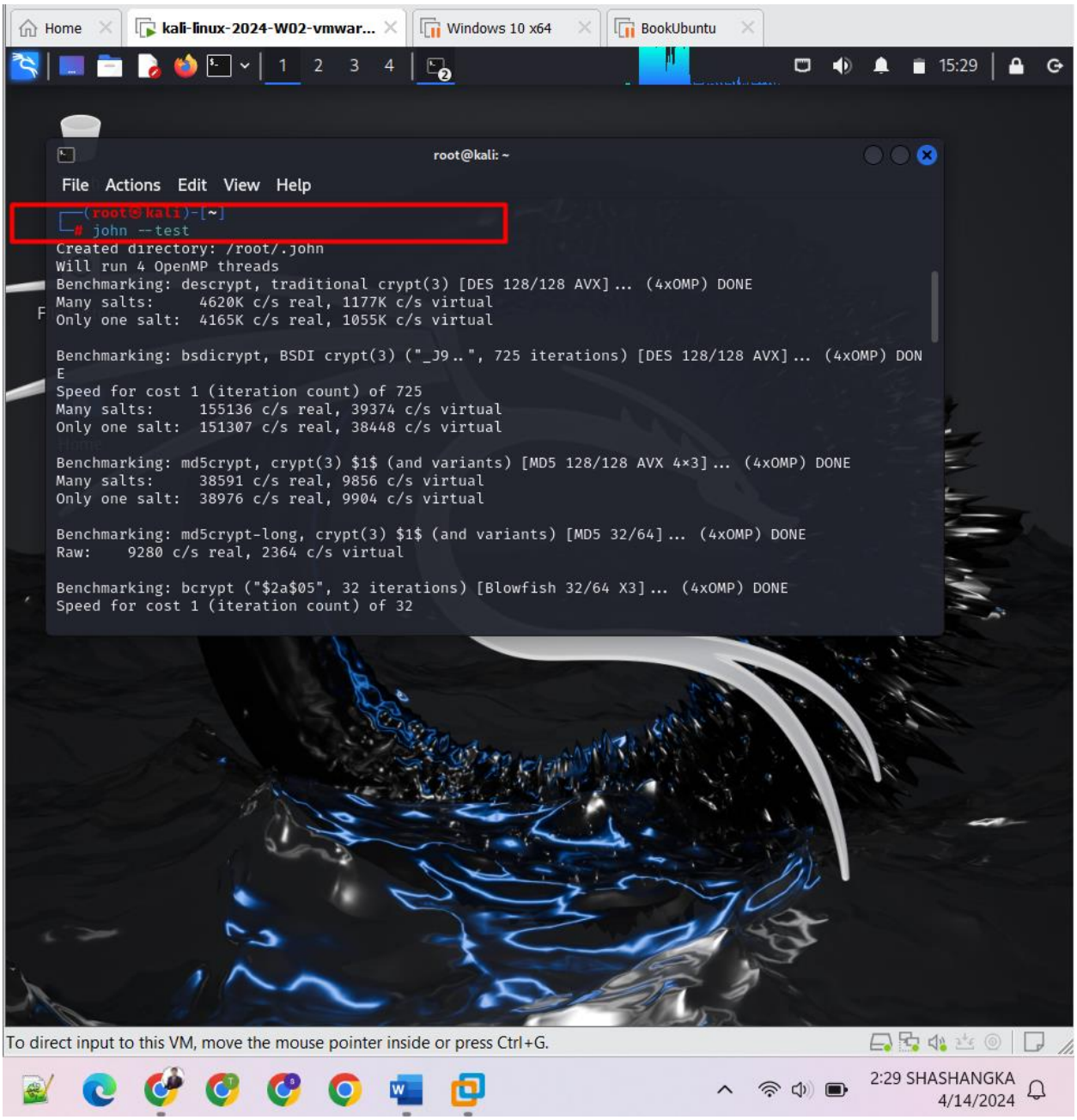
Password Cracking with John the Ripper

- **Objective: Conducted a password cracking exercise to assess the strength of user passwords.**
- **Tools Used: John the Ripper, Hashcat.**
- **Key Outcomes: Successfully cracked weak passwords by leveraging dictionary and brute-force attacks. Provided insights into best practices for creating strong, secure passwords**

Step 1:

We will start the lab to test the speed of each algorithm John will run on your particular computer hardware. Bring on a terminal on Kali Linux machine and run

john --test



Step 2: 2. Next, we will use John to crack Linux password. To save my time, I download the passwd and shadow files from the Blackboard and save them into the /tmp folder. I then use the unshadow script from John to combine account information from /etc/passwd with password information from /etc/shadow and store the result in passwdhash.txt

unshadow /tmp/passwd /tmp/shadow > /tmp/passwdhash.txt

Step 3: We now run John against the combined file.

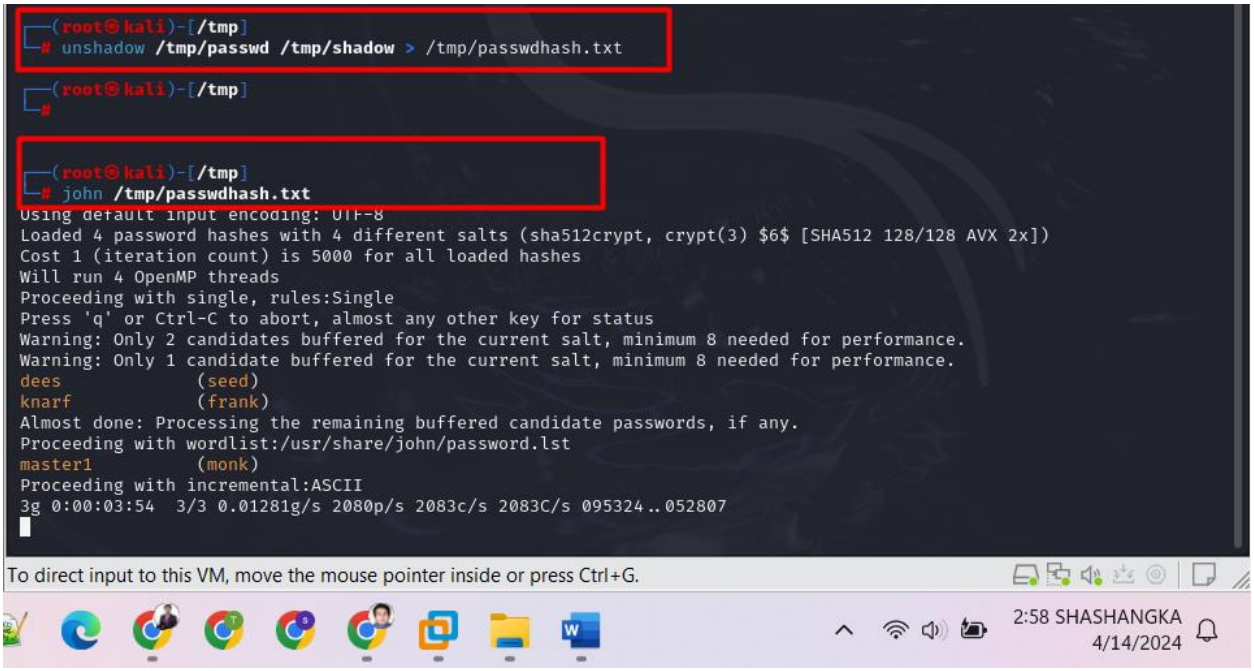
 # john /tmp/passwdhash.txt

Step 4: Next, we use --show option to review cracked hashes.

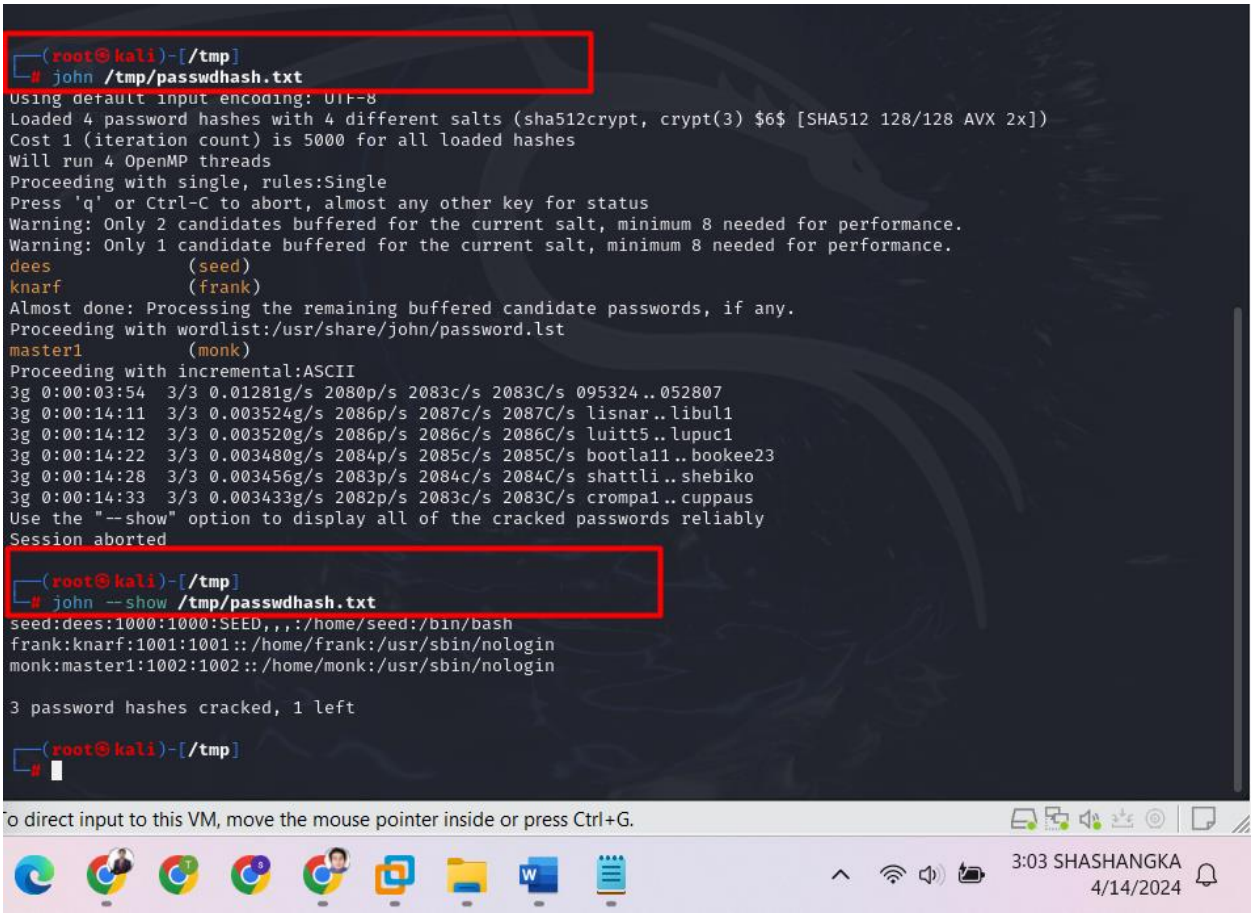
john --show /tmp/passwdhash.txt

When John cracks some password hashes, it automatically stores the cracked password hashes to a file named john.pot in a hidden directory. john under your home directory. Let's review the john.pot file by running.

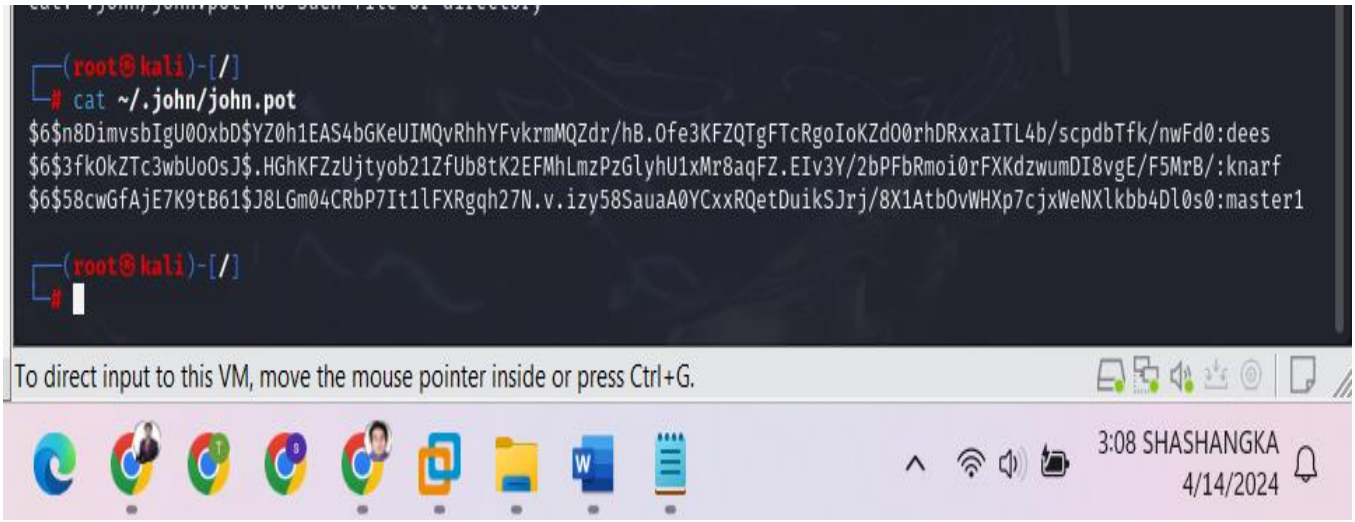
 # cat .john/john.pot



(Screenshot #1)



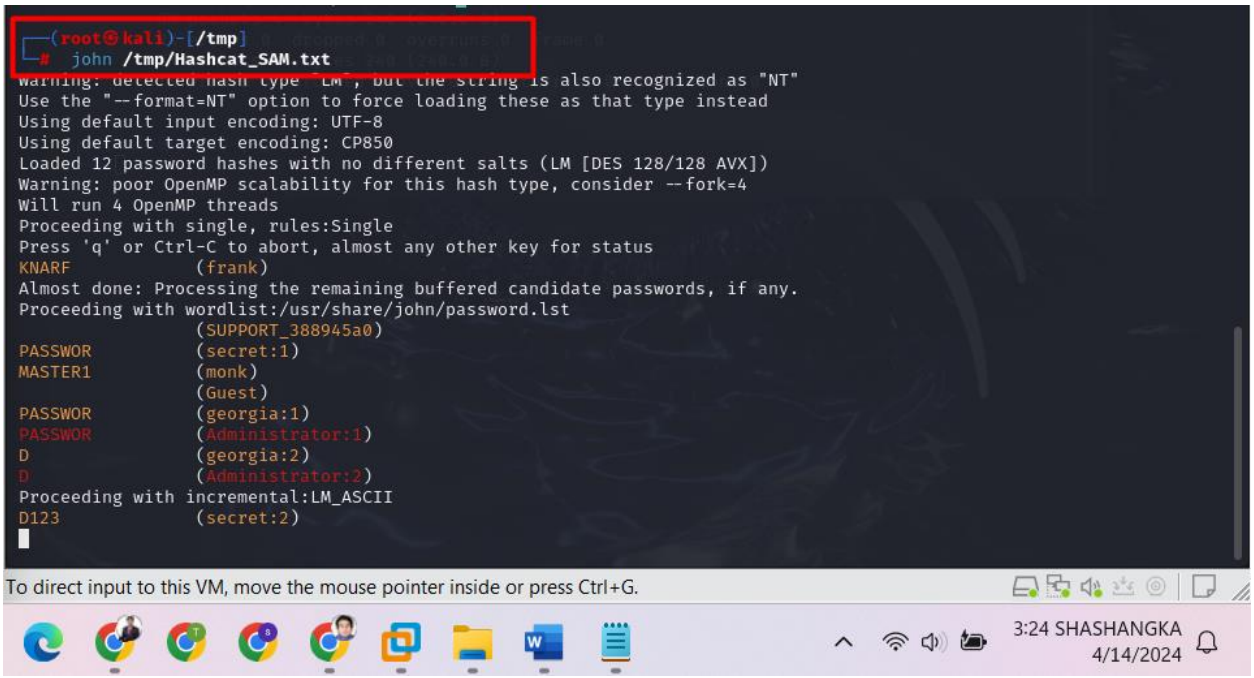
Screenshot #2



Step 5: Next, we will use John to crack the passwords dumped from the Windows XP machine. I Download the Hashcat_SAM.txt file from the Blackboard

Run John against the hash file.

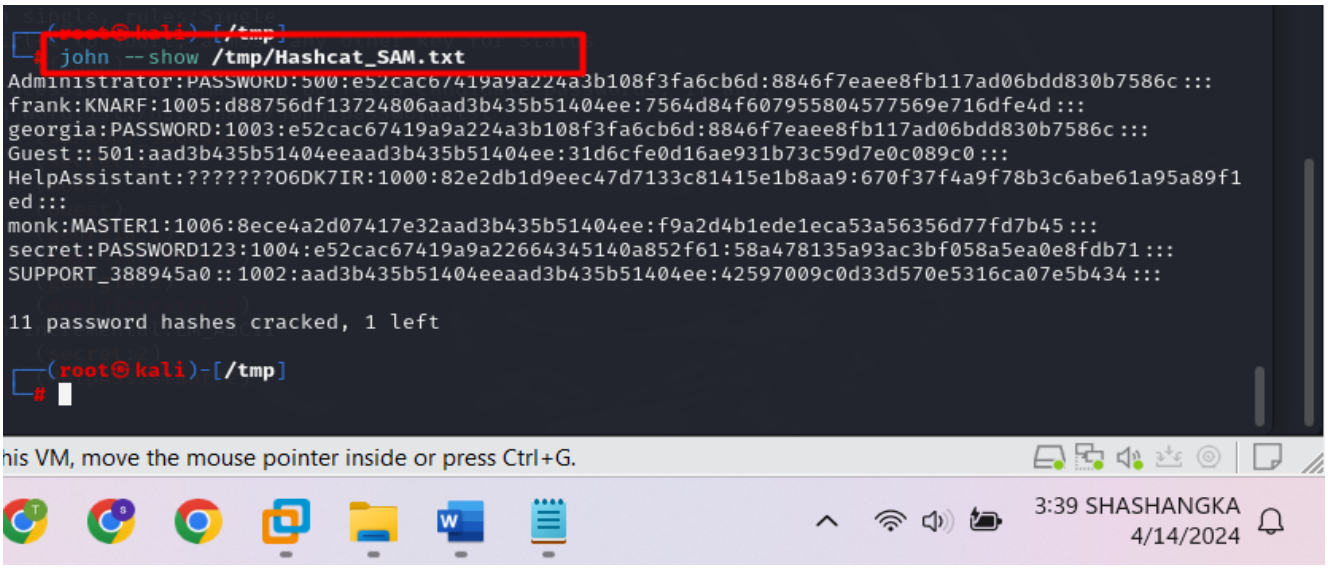
john /tmp/Hashcat_SAM.txt



Step 6:

We can run the --show option to view the full password cracked by John. This command searches inside the john.pot file for the hashes in Hashcat_SAM.txt so that it can print the full passwords associated with the users.

Screenshot #3



Step 7 :

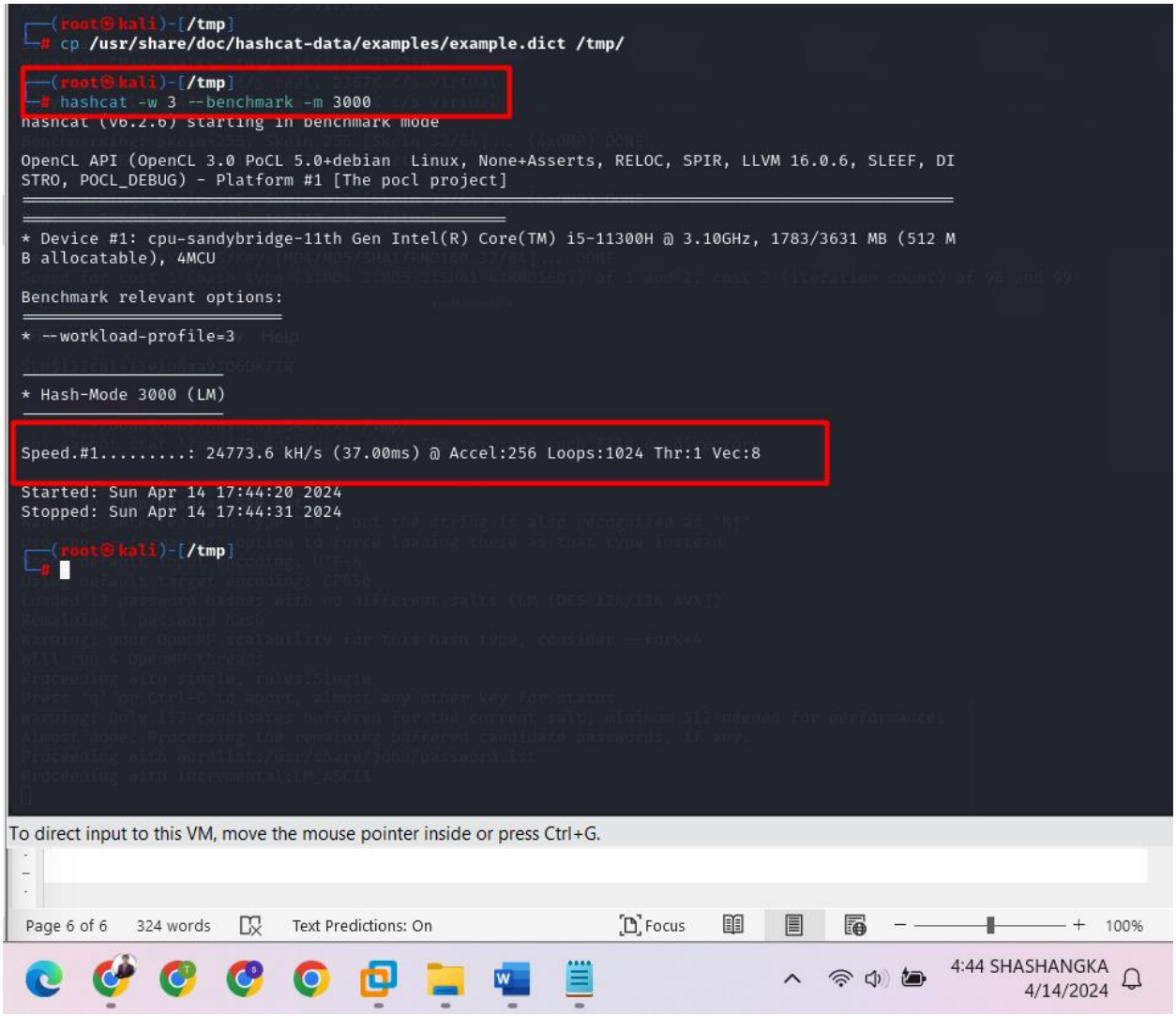
Next we will use Hashcat to crack the same hash file. First, we need to prepare the wordlist provided by Hashcat.

 # cp /usr/share/doc/hashcat-data/examples/example.dict /tmp/

Now, let's conduct some performance measures of Hashcat for some common hash algorithms. We will test the performance for cracking hash type -m 3000, also known as LM.

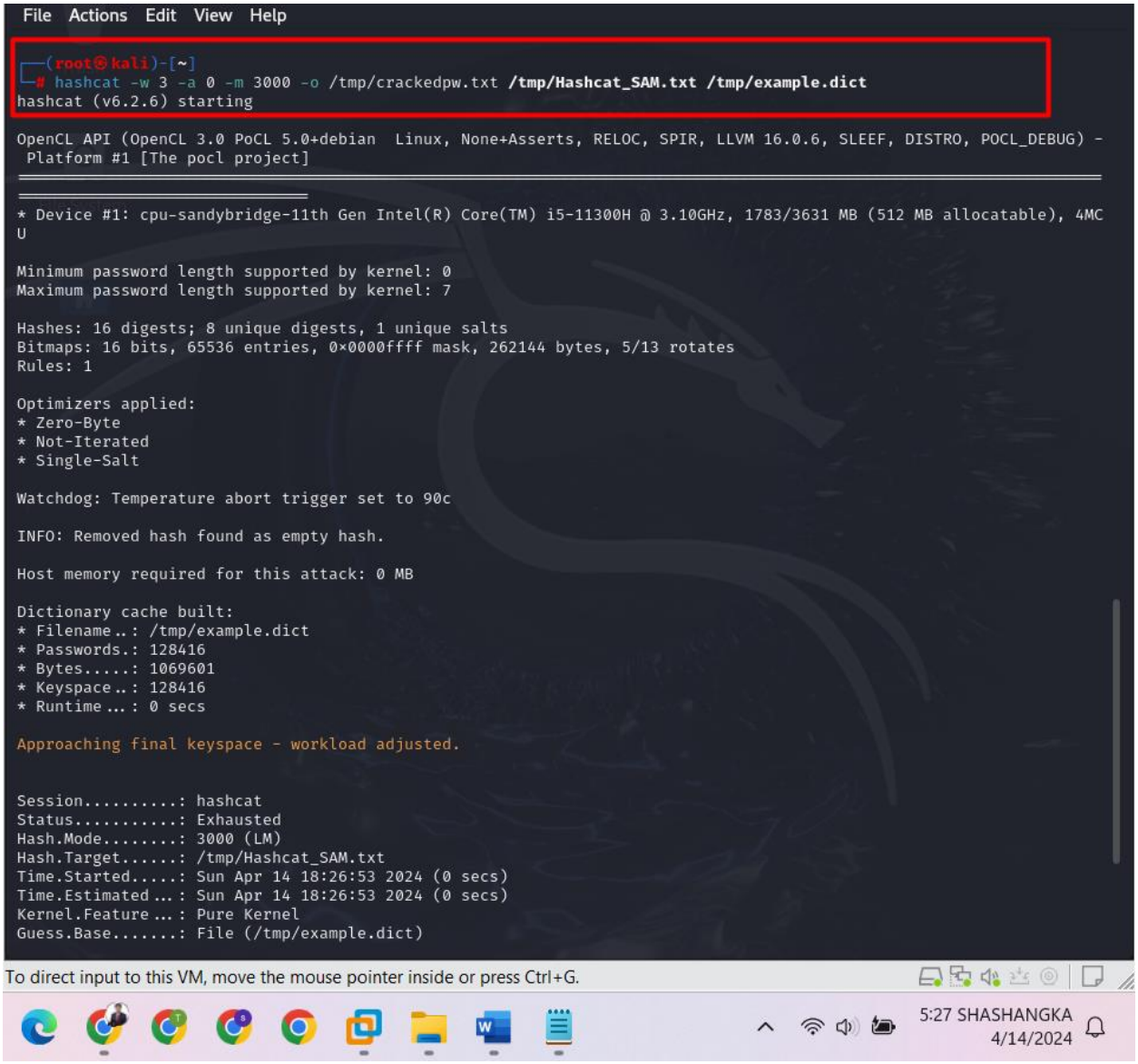
 # hashcat -w 3 --benchmark -m 3000

 # hashcat -w 3 -b --benchmark-all



Now we will crack the LM hashes (-m 3000) use the example.dict as our dictionary. The results will be stored in a file named crackedpw.txt and we will save this file in the /tmp folder.

 # hashcat -w 3 -a 0 -m 3000 -o /tmp/crackedpw.txt /tmp/Hashcat_SAM.txt /tmp/example.dic


File Actions Edit View Help
┌──(root㉿kali)-[~]
└─# hashcat -w 3 -a 0 -m 3000 -o /tmp/crackedpw.txt /tmp/Hashcat_SAM.txt /tmp/example.dict
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-sandybridge-11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, 1783/3631 MB (512 MB allocatable), 4MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 7
Hashes: 16 digests; 8 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
Optimizers applied:
* Zero-Byte
* Not-Iterated
* Single-Salt
Watchdog: Temperature abort trigger set to 90c
INFO: Removed hash found as empty hash.
Host memory required for this attack: 0 MB
Dictionary cache built:
* Filename..: /tmp/example.dict
* Passwords.: 128416
* Bytes.....: 1069601
* Keyspace..: 128416
* Runtime...: 0 secs
Approaching final keyspace - workload adjusted.
Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 3000 (LM)
Hash.Target......: /tmp/Hashcat_SAM.txt
Time.Started.....: Sun Apr 14 18:26:53 2024 (0 secs)
Time.Estimated...: Sun Apr 14 18:26:53 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/tmp/example.dict)
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
5:27 SHASHANGKA
4/14/2024

Step 8:

When Hashcat finishes running, let's look at our results. First, we check the crackedpw.txt file

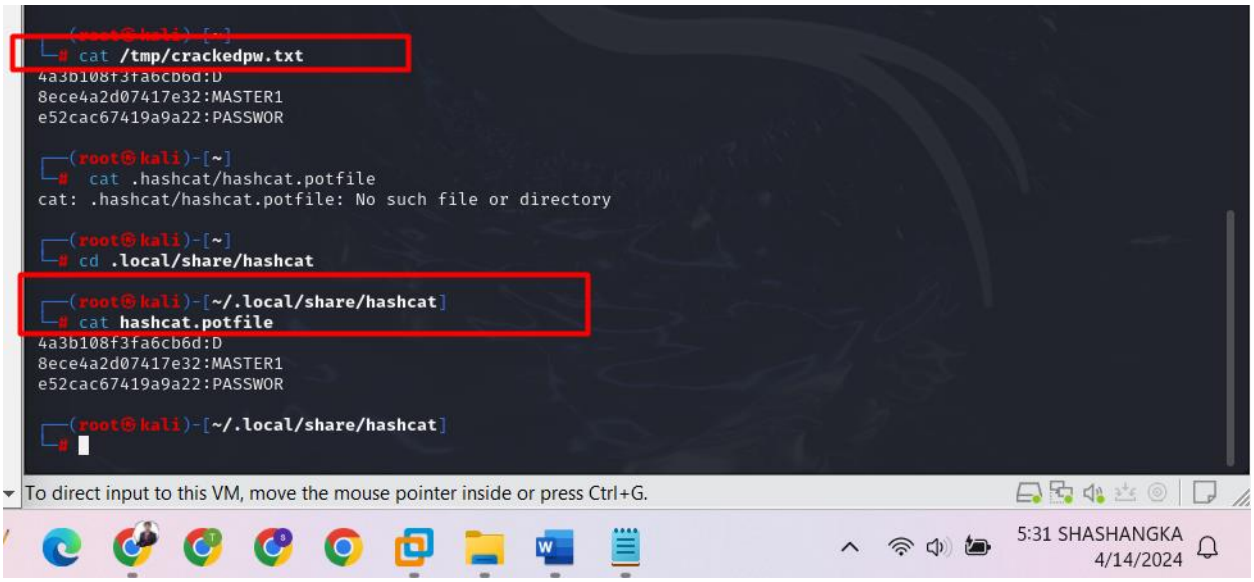
 # cat /tmp/crackedpw.txt

We can see that Hashcat successfully cracked several of our passwords, include the blank password which has a LM hash starting with aad3b43…

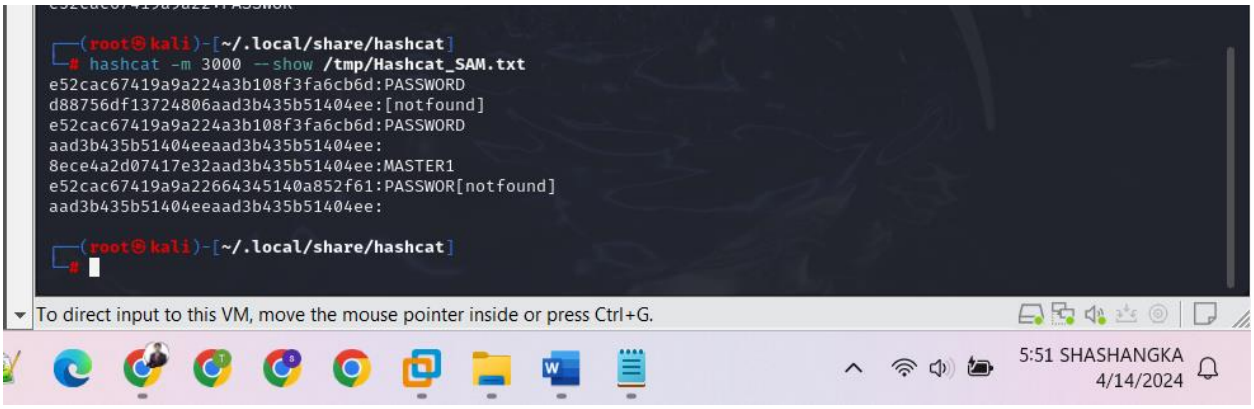
Let's also look at our potfile

 # cat .hashcat/hashcat.potfile

Screenshot 4



hashcat -m 3000 --show /tmp/Hashcat_SAM.txt



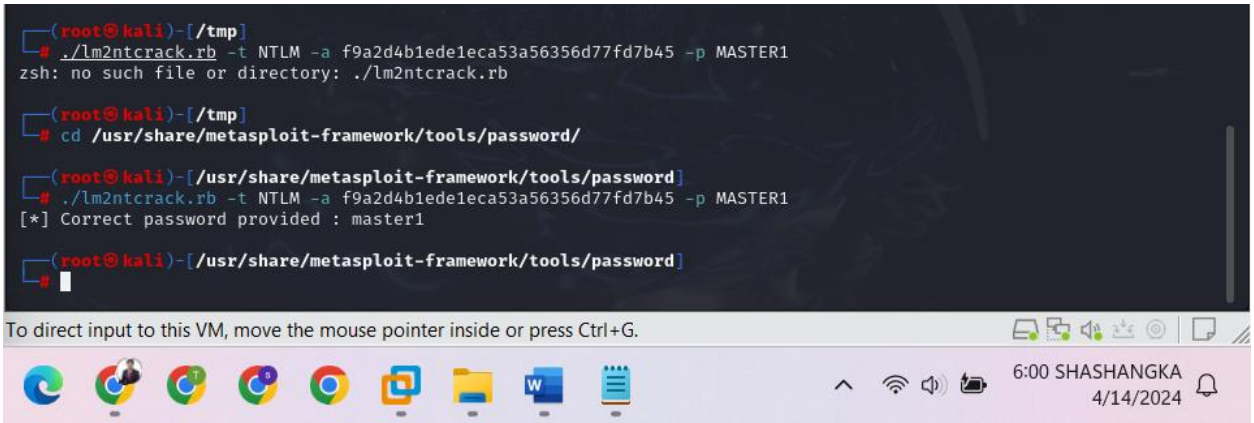
Furthermore, we can find that monk's password has been cracked (MASTER1). However, the password hash we cracked is the LM hash. The cracked password is all in CAPs. Unfortunately, such password cannot be used in the environment as it does not have the correct case. We can use a ruby script to obtain the correct case for the password.

First, we need to change directory to where the script is located by typing.

 # cd /usr/share/metasploit-framework/tools/password/

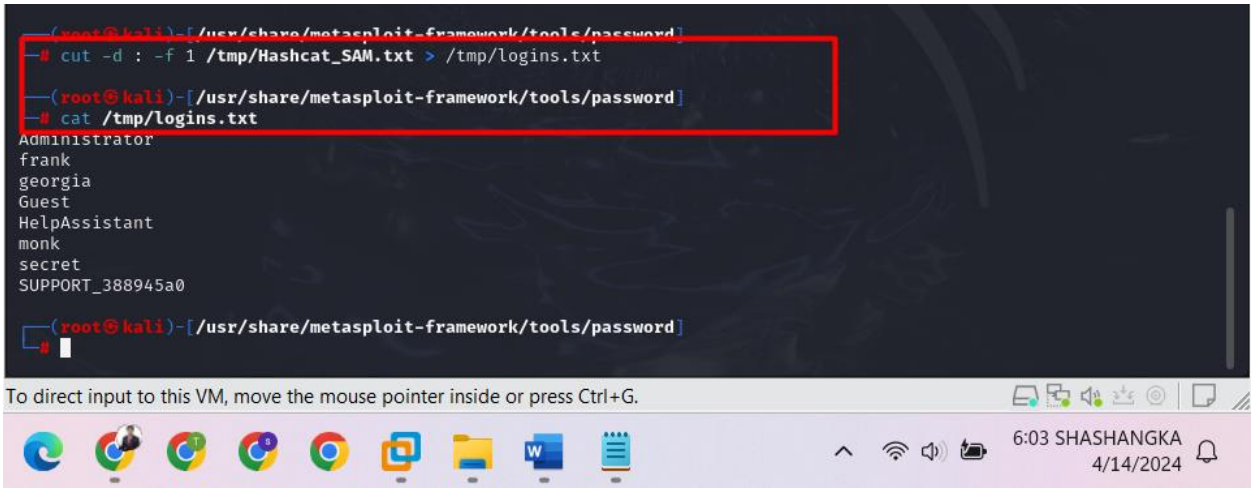
After that, run the lm2ntcrack.rb script. Secret's password was cracked correctly.

. # ./lm2ntcrack.rb -t NTLM -a monk's_NT_Hashes -p MASTER1



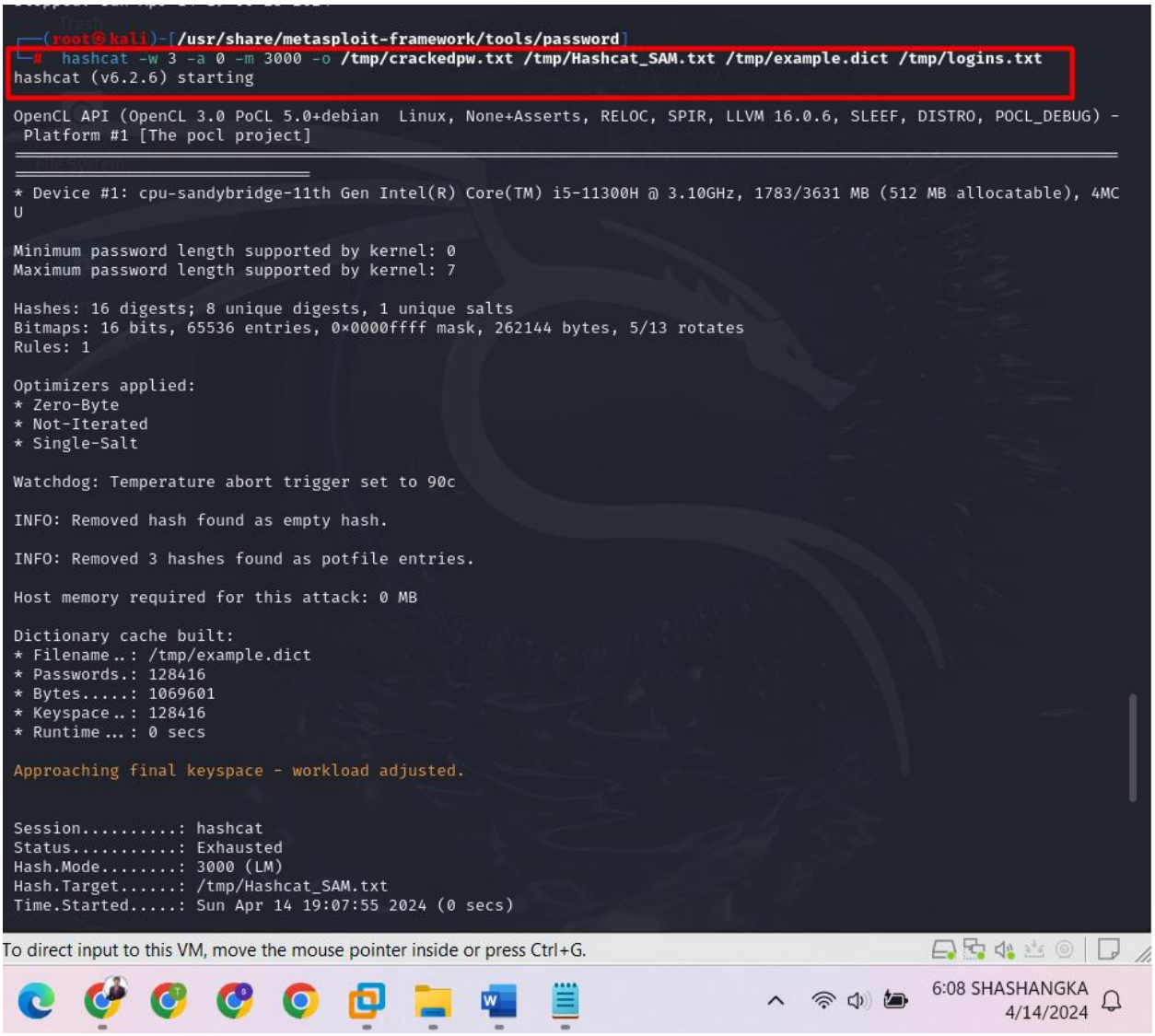
We can see the login name is the first item in this colon delimited file. We will extract the login names by using the cut command. We save the results in a file called logins.txt under /tmp folder.

 # cut -d : -f 1 /tmp/Hashcat_SAM.txt > /tmp/logins.txt

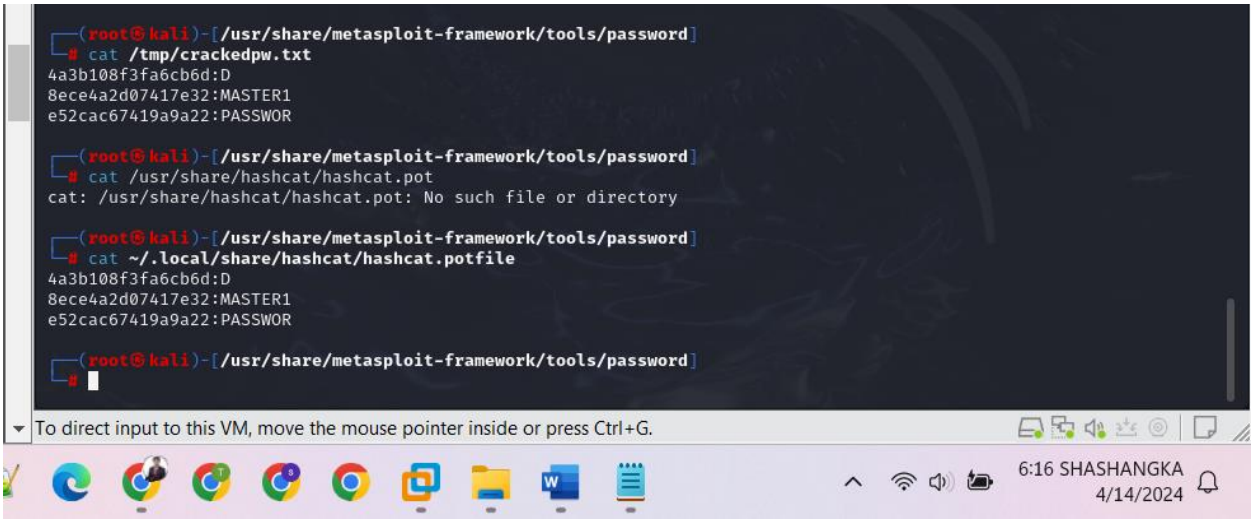


Next, we will re-run Hashcat by adding the logins.txt as a second dictionary in the command line. Hashcat conveniently allows us to append multiple dictionary files to its command line.

 # hashcat -w 3 -a 0 -m 3000 -o /tmp/crackedpw.txt /tmp/Hashcat_SAM.txt /tmp/example.dict /tmp/logins.txt



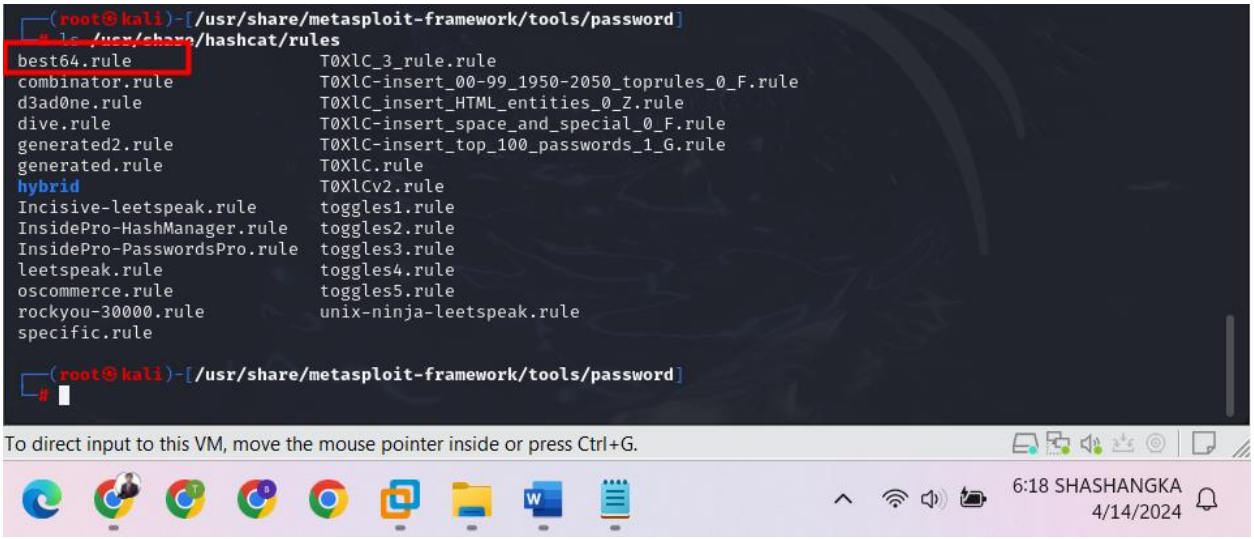
After Hashcat finishes running, I look at the crackedpw.txt and hashcat.potfile one more time. Unfortunately, we still did not crack the password for the frank account.



Kali stores all the Hashcat rules inside the **/usr/share/hashcat/rules** folder.

We Review the available rules by typing

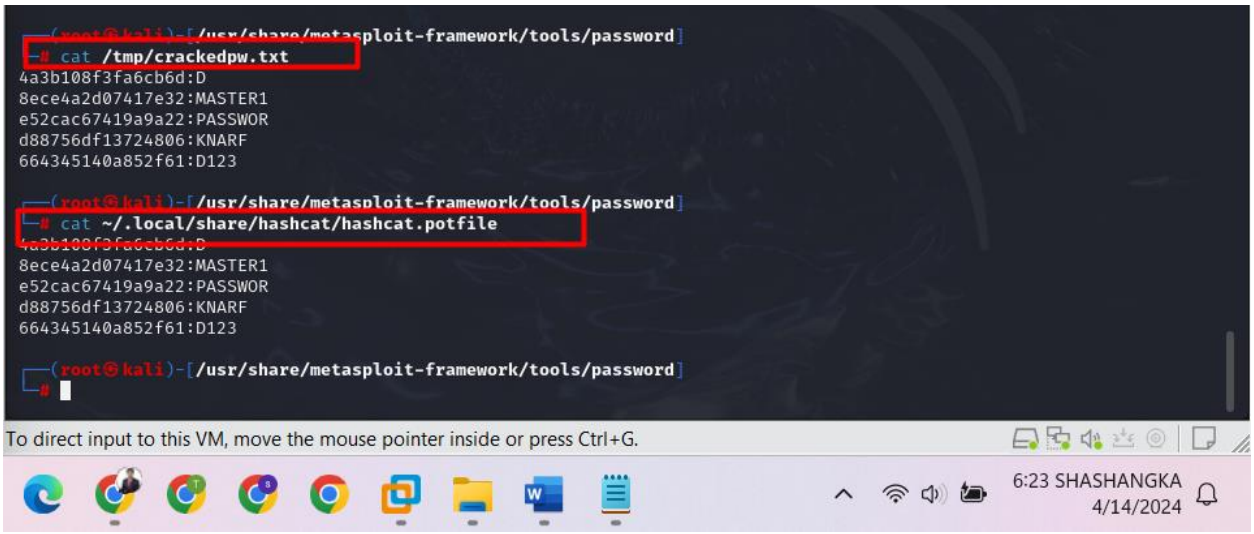
 # ls /usr/share/hashcat/rules



Let's crack the hash file one more time by specifying the rule using the -r option.

 # hashcat -w 3 -a 0 -m 3000 -o /tmp/crackedpw.txt /tmp/Hashcat_SAM.txt /tmp/example.dict /tmp/logins.txt -r /usr/share/hashcat/rules/best64.rule

Finally, we successfully crack the password for the frank account. By doing this lab, we learn that adding a login name file to our dictionary is useful with Hashcat, and applying a rule can be even more helpful.



Step 9:

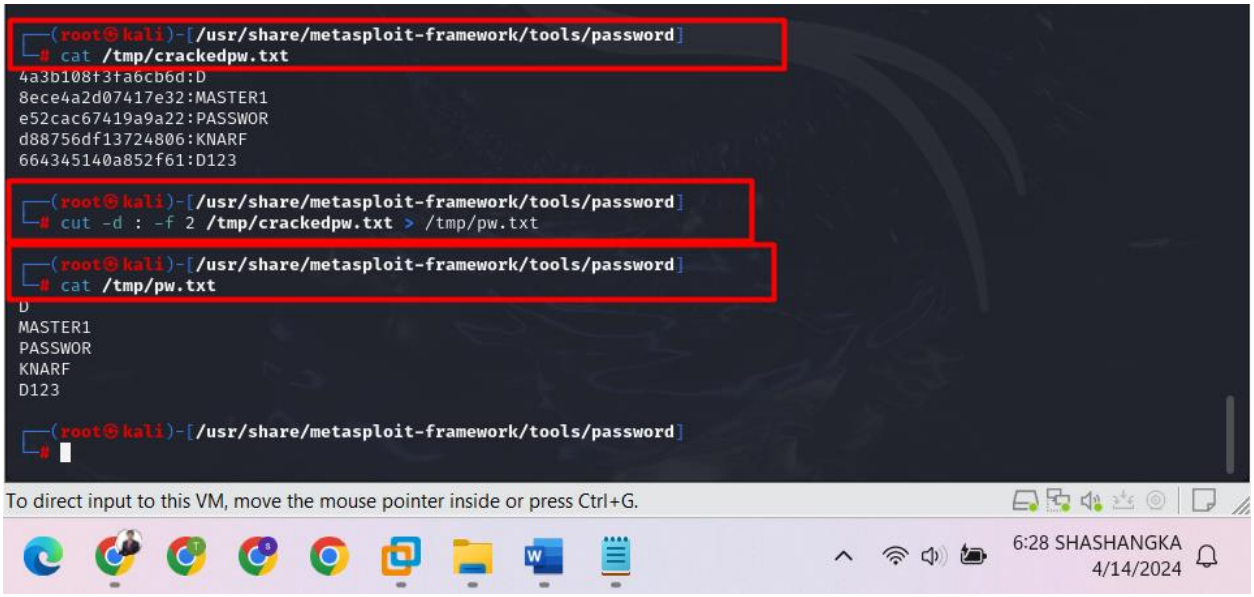
Next, we will use Hashcat to crack some Linux hash files. We will use the shadow copy from step 2.

Let's review the cracked passwords stored in the crackedpw.txt file.

 # cat /tmp/crackedpw.txt

We can see that the cracked passwords are stored as the second item in this colon delimited file. We will exact the cracked passwords by using the cut command. We save the results in a file called pw.txt under /tmp folder

 # cut -d : -f 2 /tmp/crackedpw.txt > /tmp/pw.txt



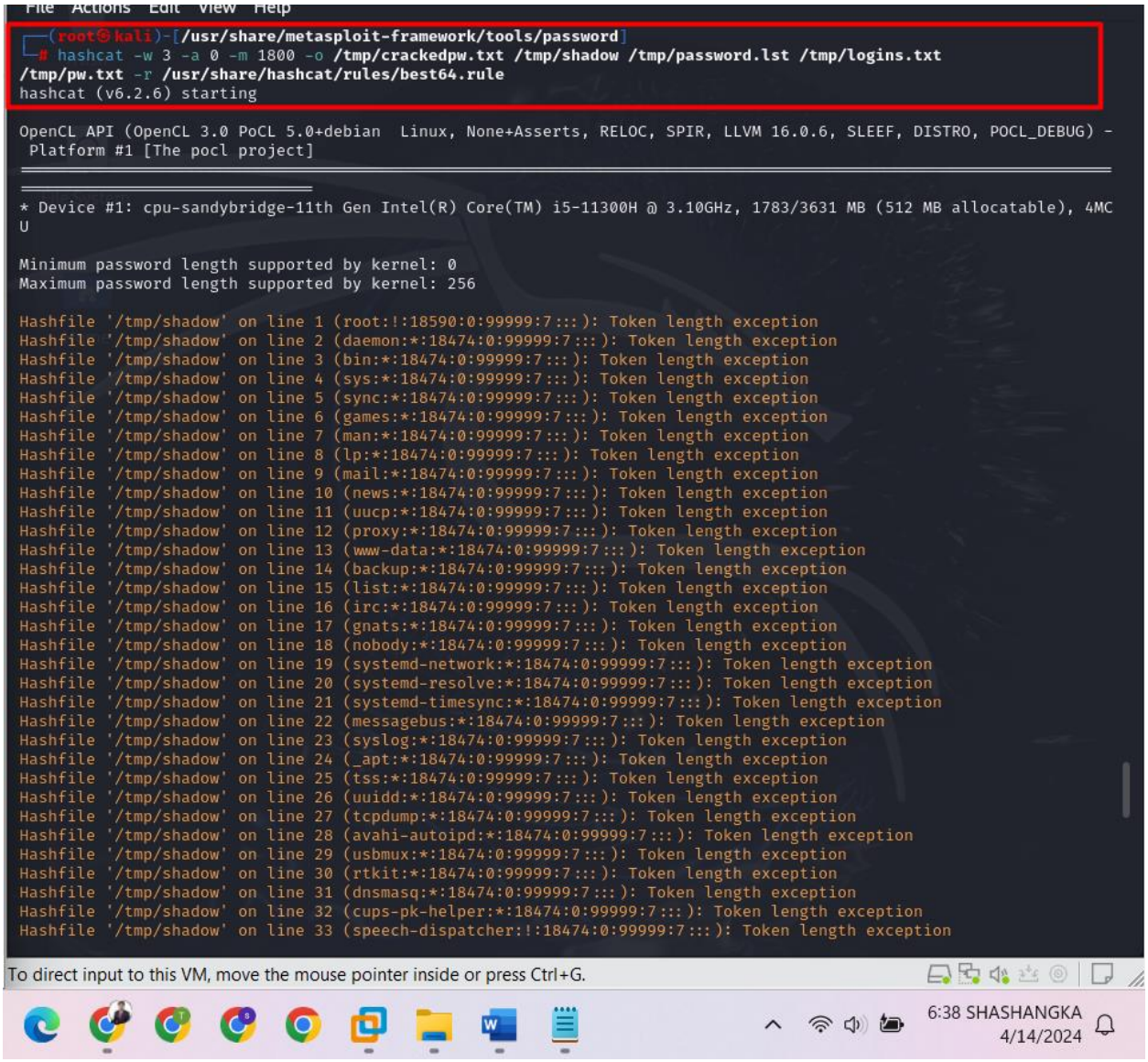
Let's copy the password.lst file to the /tmp folder.

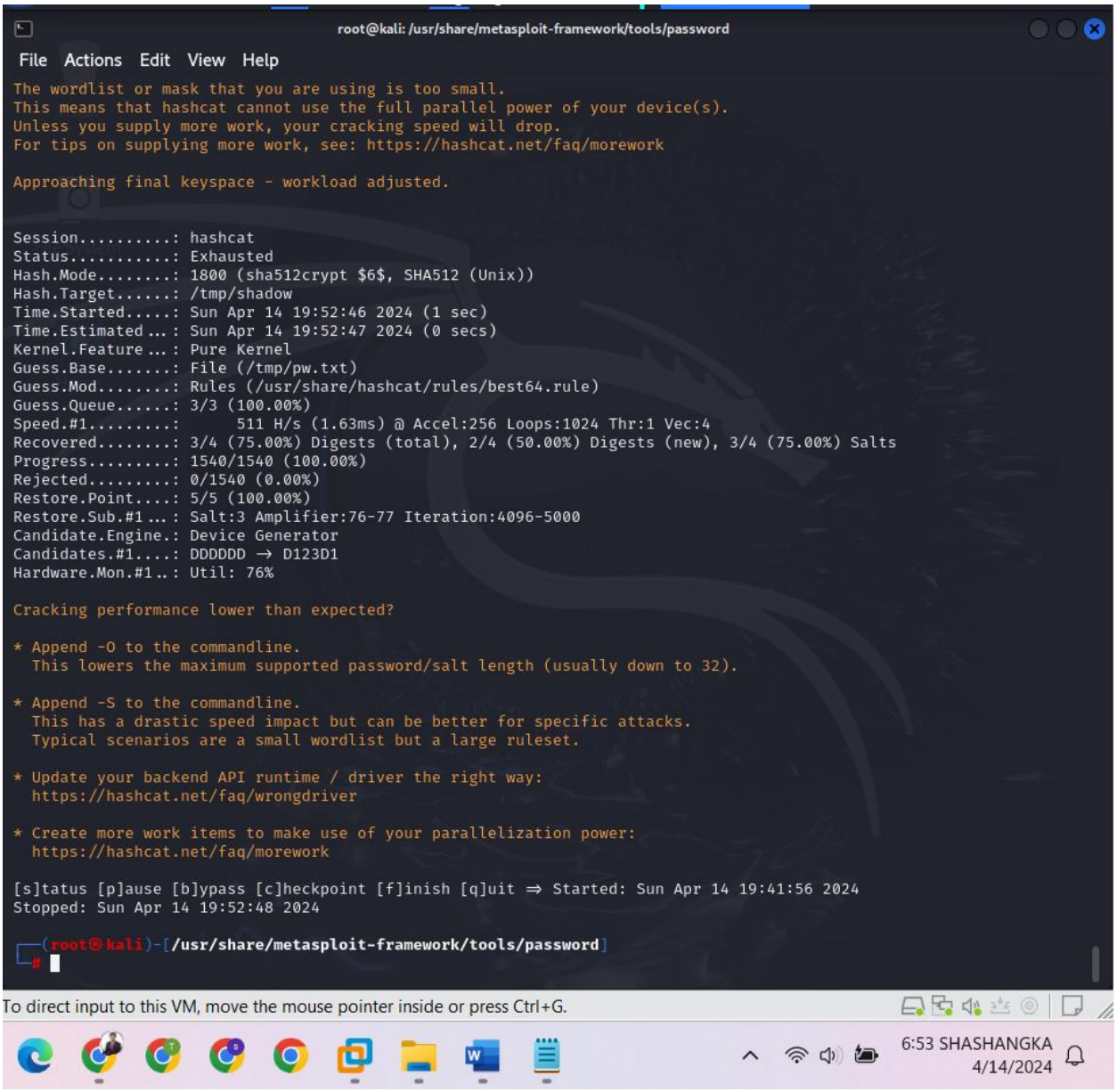
 # cp /usr/share/john/password.lst /tmp/

Since Linux use salted SHA512 for its password hashes. We will use hash type of 1800 when we run the Hashcat

 # hashcat -w 3 -a 0 -m 1800 -o /tmp/crackedpw.txt /tmp/shadow /tmp/password.lst /tmp/logins.txt /tmp/pw.txt -r /usr/share/hashcat/rules/best64.rule

As Hashcat runs, you can see that it is parsing the file to look for the hash type of 1800. When it sees a line without such hash, Hashcat will move on. When it sees a line with the 6 hash, it will try to crack it. When Hashcat runs, you can type s key to get the status. It takes a few minutes for Hashcat to finish running.


File Actions Edit View Help
(root㉿kali)-[/usr/share/metasploit-framework/tools/password]
hashcat -w 3 -a 0 -m 1800 -o /tmp/crackedpw.txt /tmp/shadow /tmp/password.lst /tmp/logins.txt /tmp/pw.txt -r /usr/share/hashcat/rules/best64.rule
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-sandybridge-11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, 1783/3631 MB (512 MB allocatable), 4MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Hashfile '/tmp/shadow' on line 1 (root:!:18590:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 2 (daemon:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 3 (bin:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 4 (sys:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 5 (sync:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 6 (games:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 7 (man:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 8 (lp:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 9 (mail:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 10 (news:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 11 (uucp:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 12 (proxy:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 13 (www-data:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 14 (backup:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 15 (list:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 16 (irc:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 17 (gnats:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 18 (nobody:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 19 (systemd-network:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 20 (systemd-resolve:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 21 (systemd-timesync:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 22 (messagebus:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 23 (syslog:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 24 (_apt:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 25 (tss:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 26 (uuidd:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 27 (tcpdump:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 28 (avahi-autoipd:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 29 (usbmux:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 30 (rtkit:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 31 (dnsmasq:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 32 (cups-pk-helper:*:18474:0:99999:7:::): Token length exception
Hashfile '/tmp/shadow' on line 33 (speech-dispatcher:!:18474:0:99999:7:::): Token length exception
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
6:38 SHASHANGKA
4/14/2024



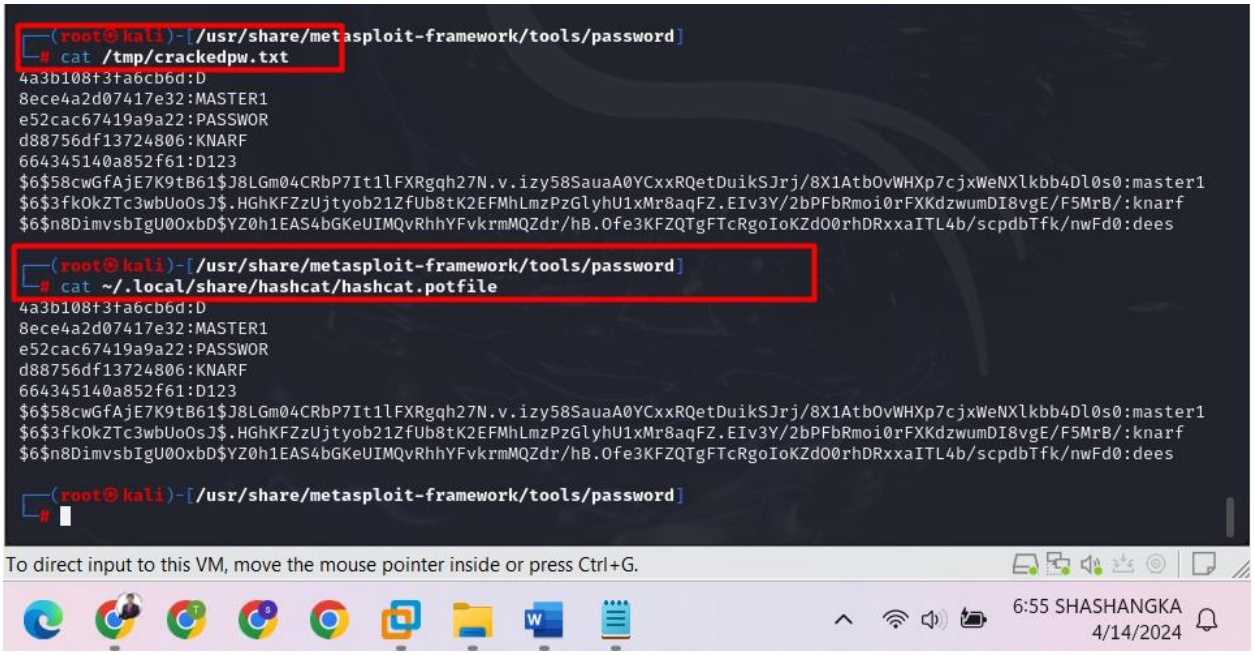
As Hashcat runs, you can see that it is parsing the file to look for the hash type of 1800. When it sees a line without such hash, Hashcat will move on. When it sees a line with the 6 hash, it will try to crack it. When Hashcat runs, you can type s key to get the status. It takes a few minutes for Hashcat to finish running.

Let's review the results.

 # cat /tmp/crackedpw.txt

 # cat .hashcat/hashcat.potfile

Screenshot 5



From the results, we can find out Hashcat successfully cracked the passwords for frank and monk. However, it was not able to crack the password for susan

. Now that we have finished this lab, let's shred the various copies of password files we left on the systems. Shred overwrites the file with alternating zeros and ones three times so that they cannot be recovered. The --remove option makes shred delete the file when it is done shredding it.

 # cd /tmp

 # shred --remove passwd

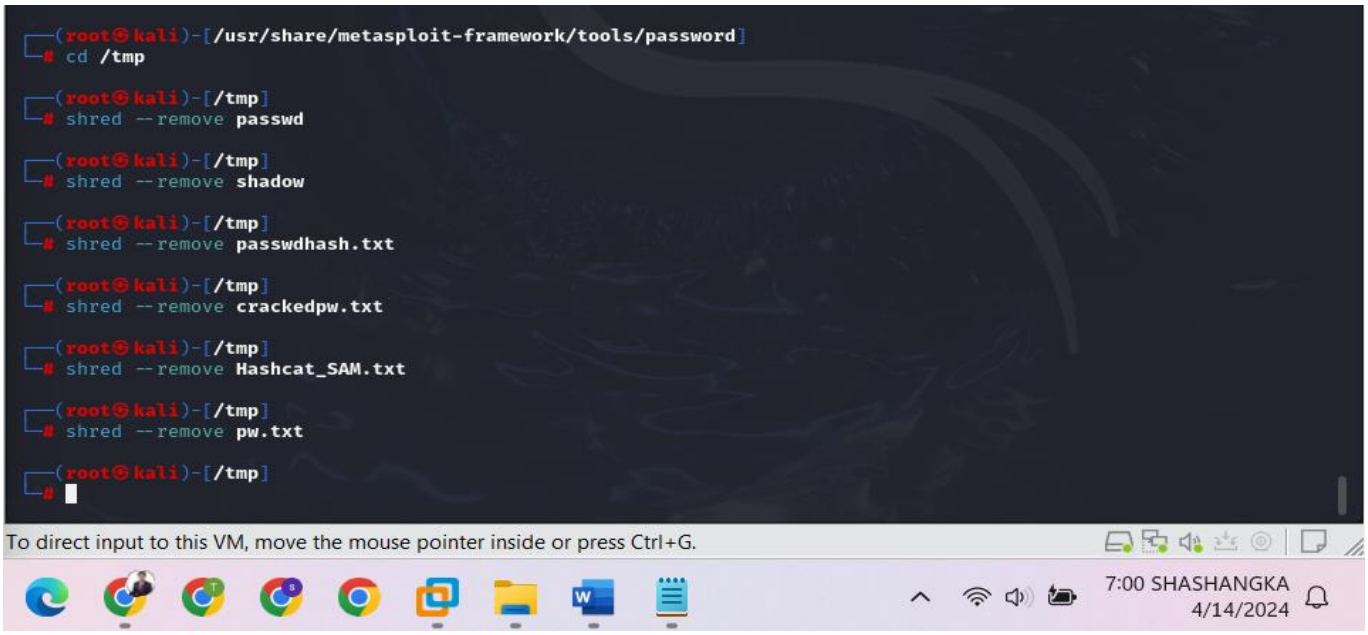
 # shred --remove shadow

 # shred --remove passwdhash.txt

 # shred --remove crackedpw.txt

 # shred --remove Hashcat_SAM.txt

 # shred --remove pw.txt


┌──(root㉿kali)-[/usr/share/metasploit-framework/tools/password]
└─# cd /tmp
┌──(root㉿kali)-[/tmp]
└─# shred --remove passwd
┌──(root㉿kali)-[/tmp]
└─# shred --remove shadow
┌──(root㉿kali)-[/tmp]
└─# shred --remove passwdhash.txt
┌──(root㉿kali)-[/tmp]
└─# shred --remove crackedpw.txt
┌──(root㉿kali)-[/tmp]
└─# shred --remove Hashcat_SAM.txt
┌──(root㉿kali)-[/tmp]
└─# shred --remove pw.txt
┌──(root㉿kali)-[/tmp]
└─#
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
7:00 SHASHANGKA
4/14/2024