☠️ LLMNR/NBTNS Poisoning

Intro

In the last session **(🔥🍗 NTLM Roasting)** we saw that we are able to intercept some packets that are sent in that authentication using ***NTLM***, we can perform a Roasting technique, crack it and ***obtain the password*** that was use it to cypher the challenge.

But the problem is that we have to stay in the middle of the infrastructure or have to capture, for any way, that authentication.

What we are going to see in this section is how, **using an external machine** *that is not even linked to the domain*, capture **those *NTLM*authentication packages.**

To do that, we have to understand how two protocols on windows work to resolve names.

When we use a name to consult a shared folder or consume some service on the network, there has to be ***some piece*** in the **Domain Controller** or in the **infrastructure** that is responsible for solving domain names because the communication below goes at the IP address level. So someone has to solve the domain name, same that's happened with a URL or a name of a website.

This domain name will normally be responsible for resolving the **DNS** that is in the **domain controller**. So when someone in the domain tries to access a shared resource through a name and not an IP address, the domain name of the machine to which we are making the request will be resolved.

But, *What would happen if we put a **domain name that does not exist in the DNS**?* **First** what windows will try to do is to resolve the name by DNS, but it won't solve it because the name does not exist. So, since it couldn't resolve it through DNS ***it will try to resolve it through other Windows proprietary protocols***.

The fist one is **LLMNR (Link Local Multicast Network Resolution)** that is a protocol that sends a request saying: *"This name '(machine name)' What IP address does it correspond to?"* to a ***multicast*** address ***where all the nodes on the network are***. I repeat, it sends **to all nodes in the network.**

And since they can't solve it by **LLMNR** uses another protocol , **NBNS (Net Bios Name Service),** that does more or less the same thing, the "*request"* is the same but makes the request on ***Broadcast to all nodes in the network.*** asd

With this in mind, *what would happen if from our Attack Machine, we were **monitoring** to see if any name resolution request was received in Windows and when one arrived, we **resolved the name and sent it the IP address of our Attacker machine and tried to establish an authentication with us**? And in this way it does the entire NTLM authentication process against us and **gets the challenger in plain text and the cypher challenge.*** Well, this is exactly what we are going to do with this technique

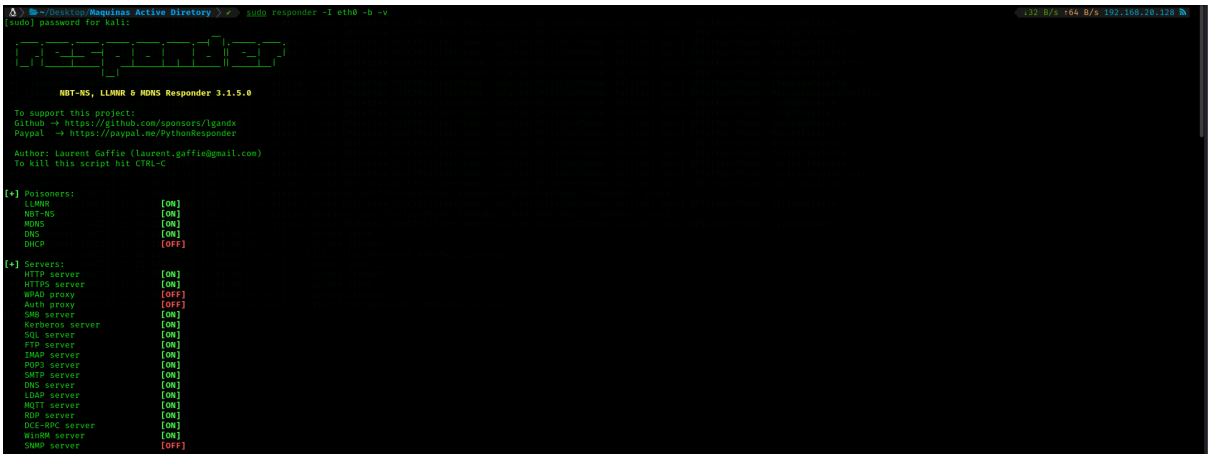
☠️☣️ LLMNR/NBTNS Poisoning

Let's see how we can do this **poisoning**, concretely, using the most popular tool for poisoning network protocols, **Responder** 🗣️.

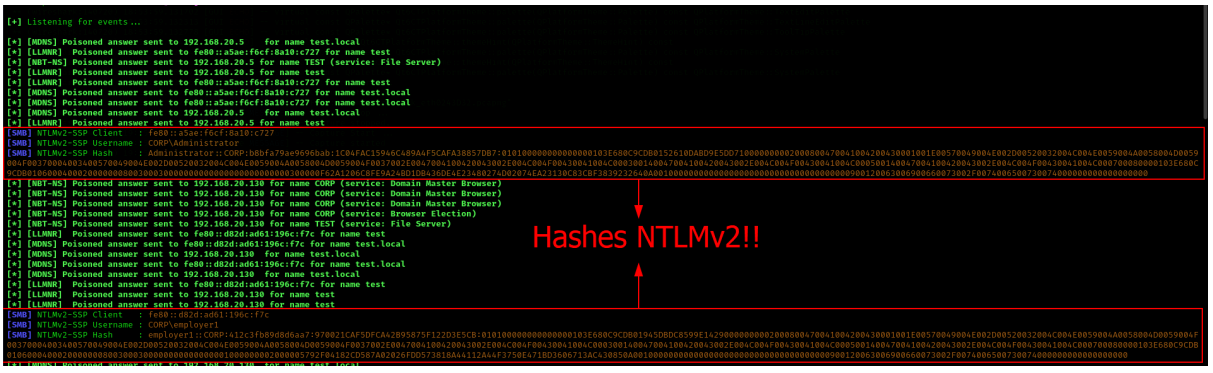
Responder 🗣️ is a tool that came by default on **Kali Linux** 🐉 and to use it on these particular case we can use these command:

sudo responder -I eth0 -b -v

With this command, the responder creates several servers for different protocols and will wait for different authentication and requests.



Now we just have to wait the moment when someone in the network infrastructure *cannot resolve a name using any of these protocols* **we will receive** the request and, as our machine is part of the network infrastructure, **we will respond by pretending to be the original node** or the original service ***so that they authenticate against us and we take their credentials,*** the hashes **NTLMv2** to crack them



We can perform this to all those users in the network that, in an infrastructure where there are **dozens** or **hundreds** of **users**, it is only a matter of time before we obtain hashes of users from that organization.

There it is another way to **get credentials without having a domain user in the domain.**
From here we can use techniques for AD and business environments