🦘🔢Over Pass The Hash / 👉🗝️Pass The Key

Intro

Once we have understanding **Pass the Hash** let's continues with a technique what is one of the more uses: ***Over pass the hash / Pass the key***

As we sew on **Pass The hash** section we can take the *hash* of an user, Admin user for example, and use it to create a new Logon session on the compromised machine and we can **replace** or **rewrite** this *hash* in the the logon session, the way this network package **NTLM** use this hash to authenticated on network services.

But as we saw, in a domain environment, normally, the network package that is used is not **NTLM** but **Kerberos**.

If we stop to think, when we were talking about Kerberos, we saw that the first thing we did to get a **TGT** was to send the **username** and a **package**, a **timestamp** *cipher with the user's private Key*. The **Private key** corresponds with the **password,** but we have seen, for example, that with **NTLM**, the password in plain text was ***discarded*** and what is saved is the **Hash of that password in memory.**

With **Kerberos,** the password is encrypted with the hash of the original password. So, when we login, using our Username and Password, what is really happened is that, our password, what we have put in plain text, is **Hashed** and that hash is **stored in memory**, in the process **lsass** and then, the authentication package **Kerberos** goes to that part of the memory and use it to get the **ticket TGT.**

The important thing here is to understand what kind of hashes are generated from the password. Concretely, **Kerberos** supports different **Kind** of **Hashes** and some of them tend to be **RC4**, **AES 128** or **AES 256.** What is the good of this? Well, the result of using a hash, like **RC4**, on the original password **is the same that generates the Authentication Package NTLM when we perform an interactive authentication.**

So, if **Kerberos** also user those hashes stored in memory to request the **TGT and other services,** one thing that we can do is exactly what we saw on **Pass The Hash,** but this time overwriting the hash that is in that area of the memory where it is going to be consulted by the **Kerberos authentication package.**

And this is exactly what **Overpass The Hash** is all about and Now we are going to see why it's Called **Overpass The Hash** and what the differences are with **Pass The Key.**

🦘🔢Over Pass The Hash / 👉🗝️Pass The Key

Now that we understand the theory behind **Overpass the Hash**, let's move on to the practical part. In the following demonstration, we will use Kali Linux and Impacket tools to obtain a **Ticket Granting Ticket (TGT)** using an **NTLM** hash, but we also can use **RC4** or **AES** if they are the hashes that we've dumped. This process will allow us to authenticate to

Kerberos services without knowing the clear-text password. We can do that using **Impacket** with the command:

impacket-getTGT domain.local/user -hashes :*"theNTLMhashcaputerd"*

Let's see how it works in the following screenshot:

And there it is, we have sent the request, ***created and we have saved*** the **TGT** of the user, administrator user in this case, on a file. This file just don't have the **TGT** but it also has the **Session Key** because we have used the **NTLM** hash of that user, which is what the initial *timestamp used in pre-authentication is encrypted with*, but it is also what the **session key that the AS** responds to us is encrypted with. For that reason we have the **TGT** but also the **session key** on that file.

And now that we ca get de **TGT** of an user and his **Private Key** and, even if it is encrypted in hash format, on next sections (🗣️ 🎟️**ASK-TGT/TGS)** we will see what we can do with this.