📇 Authorization and Authentication on Windows Systems

Intro

In this lesson we are going to talk about Authorization and Authentication Mechanisms on Windows Sistem.

The first thing we see is the classic graphic interface of login that is going to ask us for our credentials. We put *User and Password,* that information is provided to a module of the OS called **LSA (Local Security Authority)**. This module receives the *credentials* and receives the ***authentications packet that must be used***.

What is the *authentication packet*? It is the authentication mechanism that we are going to use depending on the type of user we use for authentication. For example, if we are using a **domain user** and we are in **a domain infrastructure**, the network packet is going to be **Kerberos.** So this **Local Security Authority** is going to call the library **Kerberos.dll** and this will trigger the whole authentication process of **Kerberos Protocol.**

In the case that we authenticate with a **local user** the Protocol that will be used is **NTLM**.

NTLM protocol works differently than **Kerberos**. It is based on the concept of **Challenge**. A **Challenge** is a piece of information that we have to Cypher, then we have to send back.

By the way, the important thing here is that if we want to authenticate with a local user we are going to use **NTLM protocol** and we won contact with a *domain controller* that makes the request to the **NTDS** Database, where are the users credentials in the domain.

In **NTLM** the database that will stored the credentials will be **SAM Database (Security Account Manager)**

🚪 LSA Logon Session

When a User authenticate correctly on a windows machine the *authentication package,* which can be **Kerberos (kerberos.dll)** or **NTLM (Msv1_0.dll),** what it are going to create is ***a logon session*** and it going to return information to the **Local Security Local (LSA)**. In this way, a session login is created associated with the user who authenticated on the computer and that contains certain credentials that have been previously placed.

The **LSA** module uses that information to create an **access token.**

This token includes, among other things, includes a unique identifier for the Logon Session Called **session identifier** or **session ID.** What does that mean? This means that all access tokens are going to be associated with a **Logon Session.**

In summary, the order will be as follows:

1) Introduce the credentials

2) All that information **is send it to the LSA** and **Hashed**
3) The **LSA send it to the authentication package** that correspond (**Kerberos** or **NTLM**)
4) A **Logon Session** is created, **associated with that user**
5) And then, is created an **Access Token associated with that user**
6) The token created includes a **unique identifier** for the logon session named **session ID**

🤝 Interactive Authentication

On windows there are two ways of Authentication and this is the first that we will talk about.

The **Interactive Authentication** the user is requested to provide information for the login, **Username and Password.**

The **local authority system (LSA)** then performs interactive authentication when a user logs in through the graphical user interface.

And then, if the authentication is correctly made, a logon session is created and ***a set of credentials is saved for future reference.*** This is a ⚠️**VERY IMPORTANT QUOTE**⚠️because, if we want to consume a Network Services we **won't need to use the login credentials again**, because they **are stored in the Module LSA** and this module **reuse the credentials transparently without us having to use them again.** And is this **characteristic** that we are going to exploit then.

🏜️ Non Interactive Authentication

The **Non Interactive Authentication** can only be used after it has been made an Interactive Authentication. But, *What does **Non Interactive Authentication** mean?* Well that's what we said that was a ⚠️**VERY IMPORTANT QUOTE**⚠️ when we talk about **Interactive Authentication:**

The User, once authenticated, **won't need to use the login credentials again** at the moment to, for example, consume a network service. Instead, **it will use the credentials already used and stored in the LSA module**, that will have it hashed**.**

For that reason is named 🏜️ **Non Interactive Authentication** because is used to connect several machine and network services without having to re-enter credentials.

In the machine that a user make an action to require **Non Interactive Authentication** , like make a ***Non Interactive network session***, means that in **some place in the LSA are THE CREDENTIALS OF THIS USER STORED.**

🪙 Access Token

When a user initiates a Login, the system verifies the user's password compared to the stored information on the security database that, depending on the authentication packaging, could be **NTDS** or **SAM.**

If authenticated correctly, the system generates a **Logon Session** and an **Access Token** is associated with that logon session. Each process that runs on behalf of this user has a copy of this **Access Token.**

What is an **Access Token**? An **Access Token** is an object that **describes the security context of a process or subprocess.** That is, we have authenticated and we need to know which permission we have to access to one or other object in the Windows OS. We need to be authorized to access certain objects and the piece **that manages and enables** this is the **Access Token.**

The information of the **access token** includes **the identity and the privilege associated with the user account. asd.**

The system uses the **Access Token** to identify to the user when a subprocess interacts with the ***securable objects*** or try to perform a system task that requires certain privileges.

More interesting things about the **Access Token:**

- **Each process on a window OS has a *main token* associated with it** that is going to describe the security context of the user account associated with the process.
- By default, the system uses the **main token** when a subprocess other process interacts with a **securable object.**
- Additionally, a subprocess can ***impersonate*** a client account. And the ***Impersonate*** allows that the subprocess interacts with the ***securable objects*** through the security context of the client.
- The subprocess that ***Impersonates*** a client has a **main token** and **an *Impersonate***.