﻿﻿﻿What is authentication?

﻿Authentication is a process for verifying the identity of a person (or an object or a service). When you authenticate a person, the goal is to verify that the person is not an imposter.

Local authentication

﻿Local authentication is done using the Local Security Authority (LSA). LSA is a protected subsystem that keeps track of the security policies and the accounts that are on a computer system. It also maintains information about all aspects of local security on a computer.

Types of Active Directory

There are two types of Active Directory:

* On-Premise Active Directory (AD)
* Azure Active Directory ﻿(AAD)

Authentication on On-Premise Active Directory

A picture containing chart

Description automatically generated

On-premise Active Directory has a record of all users, PCs and Servers and authenticates the users signing in (the network logon). Once signed in, Active Directory also governs what the users are, and are not, allowed to do or access (authorization).

In an on-premise Active Directory environment the authentication can be made by using the following protocols:

* NTLM
* LDAP / LDAPS
* KERBEROS

NTLM / NTLM 2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NTLM uses a challenge-response sequence of messages between a client and a server system. NTLM  provides authentication based on a challenge-response authentication scheme. It does not provide data integrity or data confidentiality protection for the authenticated network connection.

A screenshot of a computer

Description automatically generated with low confidence

LDAP / LDAPS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The main difference between LDAP and LDAPS is that LDAPS support encryption and therefore the credentials are not sent in plain text across the network.

Another thing to keep in mind is that the Domain Controller (DC) can be considered a database of users, groups, computers and so on (contains information about objects). Using LDAP/LDAPS the user's workstation sends the credentials using an API to the Domain Controller in order to validate them and be able to log in.

The procedure is similar to the image below:

Timeline

Description automatically generated

KERBEROS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Another way to authenticate is using Kerberos. Kerberos uses [symmetric-key cryptography](https://en.wikipedia.org/wiki/Symmetric-key_algorithm) and requires trusted third-party authorization to verify user identities. The authentication process is similar to the one below:

Chart

Description automatically generated

﻿Authentication on Azure Active Directory

﻿A picture containing chart

Description automatically generated

**Azure Active Directory** is a secure online authentication store, which can contain users and groups. Users have a username and a password which are used when you sign in to an application that uses Azure Active Directory for authentication. So, for example, all of the Microsoft Cloud services use Azure Active Directory for authentication: Office 365, Dynamics 365 and Azure.

Azure Active Directory supports the following authentication methods:

* SAML (Security Assertion Markup Language)
* OAUTH 2.0
* OpenID Connect

SAML (Security Assertion Markup Language)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Security Assertion Markup Language (SAML) is a type of Single Sign-On (SSO) standard. It defines a set of rules/protocols that allow users to access web applications with a single login. This is possible because those applications (referred to as “Service Providers”) all trust the systems that verify users’ identities (referred to as “Identity Providers”).

Service Providers - These are the systems and applications that users access throughout the day.

Identity Providers - This would be the system that performs user authentication.

OAUTH 2.0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

OAuth 2.0 is a standard that apps use to provide client applications with access.

OAuth 2.0 spec has four important roles:

* The authorization server, which is the server that issues the access token.
* The resource owner, normally your application's end-user, that grants permission to access the resource server with an access token.
* The client, which is the application that requests the access token, and then passes it to the resource server.
* The resource server, which accepts the access token and must verify that it is valid. In this case, this is your application.

OpenID Connect

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

OpenID Connect is an authentication standard built on top of OAuth 2.0. It adds an additional token called an ID token.

For that, it uses simple JSON Web Tokens (JWT). While OAuth 2.0 is about resource access and sharing, OIDC is all about user authentication

Source : [TryHackMe | Intro to Windows](https://tryhackme.com/room/intro2windows)