**OAUTH**

**The challenges with authentication and authorization:**

With earlier model of implementing security with authentication and authorization the UN/PW transmit over network leads to risk of theft.

UN/PW stored in DB and encrypted and validated in application also increase the vulnerability of any injection.

Scalability of the application will also open multiple end (when broken into micro services) point raising security risk.

When application broken down into micro services how to communicate with multiple components with same login/session and without transmitting UN/PW between multiple end points which also cause performance issue (This was not a problem in monolith application where there was only one endpoint and latency was low).

To address all these issues we need Tokens

**Token:**

Sharing credentials over network could make it vulnerable, anyone who breaks our network and also from persons inside our network like other employees and contractors.

So one way is encode the user details (Basic/BASE64) and pass it and each endpoint will authenticate the user details perhaps with some directory service/Identity provider LDAP.

But this method doesn’t scale well as each services needs to talk with identity provider to get the details and also reduce performance and increase risk of hack.

To solve this once a service will talk with IDP and get a Token (is a reference to a state in IDP) and pass this token and now services can communicate with each other with token and validate that token against the IDP.

Token sent along with http header just like Basic authentication but with “Bearer” prefix

But what if the token got hacked? It’s no different than a password and if you see also doesn’t solve the scaling problem as getting checked with IDP on each hub.

So its most important to token have an expiry time.

The above token are called **By Reference Token** where the state is present in IDP and referenced in each service to validate against IDP with token reference id.

Other type of token there called **By Value Token** which will have the state inside it (like Name, Email, Scope, Expiry time etc which can be decided by us) and will be signed by IDP and each service will validate with public key. This tokens solves the scaling problem if you see it doesn’t need to contact with IDP and state is present inside the token itself. But anyone can see the content if they have access to the token. Which can solve using encrypting the content.

So what we use is a **standard format** because we don’t want different components in our application communicating with each other in different format.



So the standard token are there:

* Kerberous (1990 protocol specific)
* SAML (2002 protocol specific)
* JWT (JSON Web Token 2015 protocol agnostic)