Api handles enormous Wide range of data. One of the main concern with data provider is to provide security for the data. Data should be secure, unchanged and available for manipulation is key to any conversion on Api data

3 major methods to add security to API

1. HTTP Basic authentication
2. Api key
3. OAuth

Authentication vs Authorization

Authentication about who you are when an entity proves its identity. You are who you say you are.

Authorization: what you can do in the system. When an entity proves a right to access.

For Eg ; Driver license

It proves who you are who you say via your picture and other particulars and also proves with this card you have right to drive car, bike, geared or gearless vehicle.

Most api solution we give piece of code that authenticate the used and provides authorization information

In most of the system we have a piece of the code that does the authentication and authorization of the user. In such cases we have hybrid system.

### **HTTP Basic Authentication:**

Http user agent has to simply provide username password to prove his identity. Approach does not need any session id, cookies, Login page because it uses HTTP headers. No need of handshake

Disadvantage:

1. Process is enforced through the data cycle to SSL for security.
2. Authentication information is transmitted in open insecure lines f the request is not enforced with SSL
3. Lead to any middle man attach
4. Where in hacker can capture the credentials and copy the header info to the malicious packet
5. Even if SSL is enforced it will reduce the response time. In base form HTTP is never encrypted its just encapsulated with Base64

Use will be internal application where network is secured

### **API Keys:**

Industry standard authentication mechanism

Introduced to fix the problems of basic authentication.

User will be assigned with a unique key when he try to access the system for the first time. Indicating the this is a known user.

SO on successive request user has to send a same key for authentication.

It will speed up the process of authentication.

Also possible to purge this key so that on expiration key cannot be used anymore in the system.

Draw back:

Just as network key can be picked up anytime over the network oif the network is not secure. And can be miss used.

### **OAuth:**

OAuth combines Authentication and Authorization to allow more sophisticated scope and validity control.

Bit strange beat. It’s a combination of authentication and authorization technique

When oAuth is solely used for authentication its called pseudo-authentication.

Process:

User loges into the system

System request for authentication. Usual happens in the form of token.

User forward the request to authentication server which will either authenticates or reject the user. From authentication server token is given to the user and then for the requester.

Such tokens can be then checked by the requester at any time for validation.

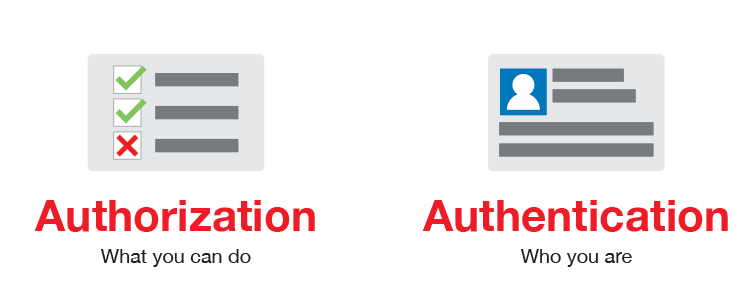
It has expiration time as well.

Authentication:

Entity proves its identity. User proves who you are who you say you are.

**Authorization** :

Entity proves a right to access the system. It proves you have right to make request.



#### 4 Most Used Authentication Methods

* Basic
* Bearer
* Digest
* OAuth  
  and others...

Authentication happens over HTTP

Basic: least recommended authentication mechanism used because of it security vulnerability.

Straight forward approach where is sender places credentials as a part of the request header.

And encoded using base 64 format so that it can be transmitted over network.

Do not need any session Cookies and Login Ui stuffs.



##### Bearer Authentication

HTTP authentication schem that uses token called bearer token. As a name suggest it grants access to the bearer carries this token.

Similar to basic authentication this can be used for communication with HTTPS over(SSL)

Client which want to access the secured resources has to send this token as a part if the request header.

Originally created as a part of OAuth2.0.

Request header looks like this.



Grant access to the resources or url Is a cryptic string usually generated by the server on login request.

#### 2. API Keys:

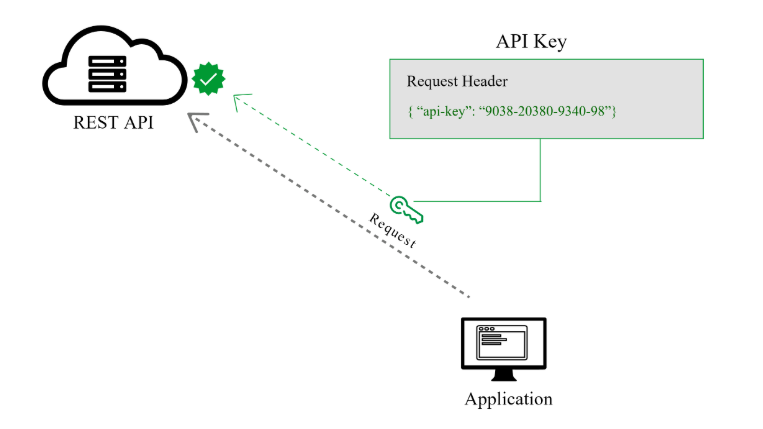
Initially introduced to overcome the disadvantage of Basic Authentication.

Widely used In the industry to protect Resources access using the Rest api. However, this is not best way to access it so far.

In this model a unique key is generated and assigned to the user when the user try to access the system for the first time. This is to indicate that the user is known to the system.

Generated based on the user’s hardware combination and IP by the sever

Next time when user try to access the system again, he need to send this key to indicated he is known to the system and he is a same user logged in some time back in past.

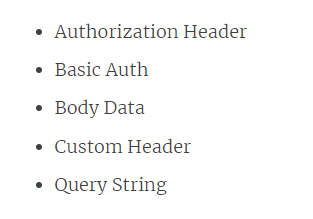


Many Api key is send as a part of URl which is not safe. Because hacker my easily intercept the request because it’s a plain text

Its always safe to send it as a part of Authorization header.

 Authorization: Apikey 1234567890abcdef.

Api Key can show up in



Use full in the scenario where the operation in the APi is limited only read operation no delete edit insert operation possible generate simple key to authorize user.

Always do security test in the web api to protect your system from the security vulnerability.

#### 3. OAuth (2.0)

Most sophisticated authentication mechanism which is a bundle of both authentication and autherization.

User 2 tokens:

1. Access token: Sent like Api key used to get access to the user data from the API, Option lay it can expire.
2. Refresh token: Optional part of OAuth, user to get a new token if the access token is expired.

OAuth is the best way to identify the personal user account and grant the access to it.

User initially logs into the system. Then system requests for the authentication usually in the form of Token. Then user forward this request to authentication server. Then server will either grant or reject the request by sending the token to the user and then to the requester. Such token can be checked any time for the authentication independent of user.

Token has limited scope and age validity.

This is a sophisticated way of authentication.

Because initially it provides a way to acquire scope. Once done it provides the access to the system.

**OAuth 2.0 Popular Flows**

The flow (also called grant types) scenarios an api client performs to get and access token from authorization server.

1. Authorization code:  The most common flow, mostly used for server-side and mobile web applications. This flow is similar to how users sign up into a web application using their Facebook or Google account.
2. Implicit:  This flow requires the client to retrieve an access token directly. It is useful in cases when the user’s credentials cannot be stored in the client code because they can be easily accessed by the third party. It is suitable for web, desktop, and mobile applications that do not include any server component.
3. Resource owner password:  Requires logging in with a username and password. Since in that case, the credentials will be a part of the request, this flow is suitable only for trusted clients (for example, official applications released by the API provider).
4. Client Credentials: Intended for the server-to-server authentication, this flow describes an approach when the client application acts on its own behalf rather than on behalf of any individual user. In most scenarios, this flow provides the means to allow users to specify their credentials in the client application, so it can access the resources under the client’s control.

#### 4. OpenID Connect

1. Acts as a identity layer on top of the OAuth2.0 protocol.

2. which enabled computing client to verify the identity of end user based on the authentication performed by the authorization server. As well as obtain some profile information of end user.

In an interop or Rest like manner

Uses Json data format for the restful api service

Support web based, mobile based, javascript based clients to request and receive the information about the authenticated client or session

OpenId has sign in flow that enables client application is perform user authentication and send some information or claims about the user such as user name password. User identity is coded in secure token called Jason web token, called Id token.

JWT token:

Open standard representing claims between two parties. JWT allows you to generate, decose verify JWT

Open Api specification;

Indicts what types of security used across the api

Number of standard authentication protocols you can pick from ach has its own strength and weekness.

Open id has discovery mechanism called open id connect discovery. When in the openId server publishes its meta data to a well known Url

<https://server.com/openid-configuration>.

Returns Jason listing of Oauth/openId end points, Supported scopes and claims. Public key is used to sig the tokens and other tokens.