**What is API:**An API, or Application Programming Interface, is a set of rules and protocols that allows different software applications to communicate with each other. It defines the methods and data formats that applications can use to request and exchange information.

**Types of API:**

There are primarily three types of APIs:

1. \*\*Open APIs (**Public APIs**):\*\* These APIs are available to developers and users with minimal restrictions. They are typically designed for public consumption and allow external developers to access certain features or data of a platform or service. Examples include APIs provided by social media platforms like Twitter or Facebook, which allow developers to integrate their applications with these platforms.

2. \*\*Internal APIs (**Private APIs**):\*\* Internal APIs are designed for internal use within a company or organization. They are not exposed to external developers and are used to facilitate communication and data exchange between different components or services within the organization's infrastructure. For example, a company might use internal APIs to connect their mobile app with their backend servers.

3. \*\***Partner APIs:\*\*** Partner APIs are shared with specific external parties or business partners under certain agreements or terms. They enable collaboration between different organizations or entities by allowing controlled access to specific functionalities or data. These APIs are often used in business-to-business (B2B) integrations or collaborations between companies. An example could be a payment gateway API provided by a financial institution to its partner merchants.

**REST API:**

A REST API, or Representational State Transfer Application Programming Interface, is a type of API that follows the principles of the REST architectural style.

It allows systems to communicate over the internet using standard HTTP methods like GET, POST, PUT, DELETE, etc., to perform actions on resources (such as data or services) identified by unique URLs.

**Statelessness**: Each request from a client to the server must contain all the information necessary for the server to understand and fulfill the request. The server doesn't store any client state between requests.

**Resource-Based:** REST APIs expose resources (e.g., data entities) that clients can interact with. These resources are identified by unique URIs (Uniform Resource Identifiers).

**CRUD Operations:** REST APIs typically support CRUD (Create, Read, Update, Delete) operations using standard HTTP methods:

GET: Retrieve a resource.

POST: Create a new resource.

PUT: Update an existing resource.

DELETE: Remove a resource.

**Representation of Resources:** Resources in a REST API can be represented in different formats such as JSON (JavaScript Object Notation) or XML (eXtensible Markup Language).

**Hypermedia as the Engine of Application State (HATEOAS):** This principle suggests that the API should provide links dynamically in the response to guide clients on how to interact with the available resources.

**API METHODS**

**GET**: The GET method is used to retrieve data from a server. When you make a GET request to a specific URL, the server returns the data associated with that URL.

**POST**: POST is used to send data to a server to create or update a resource. It's commonly used when you want to submit data to a server, such as submitting a form or adding new data.

**PUT**: PUT is used to update data on the server. It replaces the entire resource at the given URL with the new data provided in the request.

**DELETE**: DELETE is used to remove a resource from the server. It deletes the resource identified by the given URL.

**PATCH**: PATCH is used to apply partial modifications to a resource. Unlike PUT, which replaces the entire resource, PATCH is used to apply partial updates to the resource.

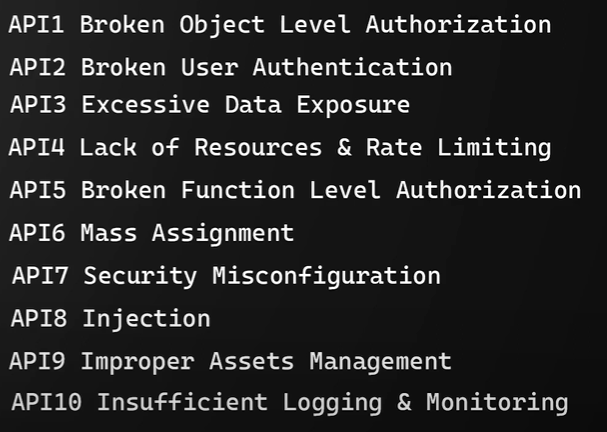
**OPTIONS**: OPTIONS is used to describe the communication options for the target resource. It specifies which HTTP methods are allowed on a particular resource, among other things.

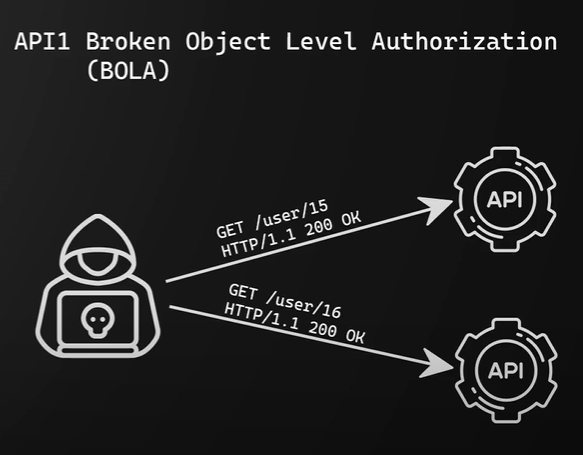
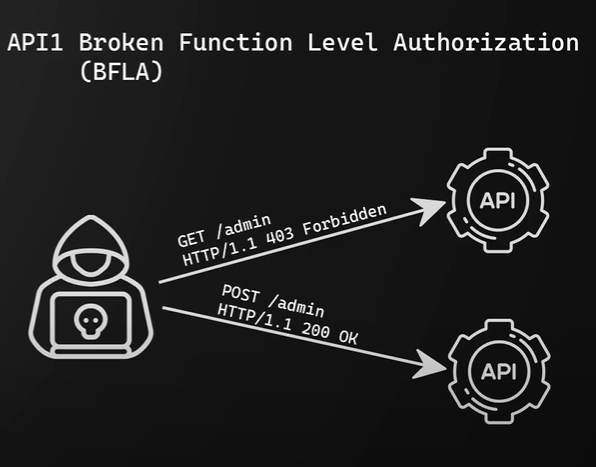
**HEAD**: HEAD requests are similar to GET requests, but instead of returning the actual data, they only return the headers that would be returned if a GET request were made to the same URL. HEAD requests are often used to check for the existence of a resource or to retrieve metadata about a resource without actually downloading the resource itself.

**RECON:**

1. WFUSS -C -Z file,usr/share/wordlist/dirb/common.txt --sc 200 <url>
2. python3 xnLinkFinder.py -i api.exoscale.com -sf api.exoscale.com -o exoscale1.txt -op param1.txt
3. fuff -u url -w common.txt

**OWASP TOP 10:**



1. **BOLA**

Broken object-level authorization (BOLA) vulnerabilities occur when a user is able to access other users' data due to the flaws in authorization controls validating access to data objects.

**Fix:** <https://www.apisec.ai/blog/broken-object-level-authorization>.

1. **BFLA**

Focuses on general API functions rather than individual API objects

FIX: It is critical that an API security tool is capable of baselining continuously, as APIs may go through a high rate of change as a result of modern development and release practices. API security solutions must be able to identify and prevent attackers or unauthorized users from accessing administrative level capabilities or unauthorized functionality.

https://owasp.org/API-Security/editions/2023/en/0xa2-broken-authentication/