FORTINET | USA

XPERTS

SUMMIT 2024

Helping you create a digitally secure future.

Mastering HTTP-based APIs

Module 02

Agenda

- Introduction to APIs
- Different Types of APIs
- API Authentication Methods
- Overview of OWASP API Top 10
- Tools for API Development and Testing

XPERTS SUMMIT 2024

Introduction to APIs

Sub header

What is an API

- Stands for "Application Programming Interface.”

- An API is a set of rules and protocols for building and interacting with software applications. It defines methods and data structures to interact with various software components.

Think of an API like a waiter in a restaurant - the link between your table (the user) and the kitchen (system). You request food (data), and the waiter ensures you get what you asked for.

What is it Good for?

- APIs serve as interfaces between different software programs, enabling them to interact without user intervention.

- Streamlines processes by allowing new integrations with existing applications, services, or platforms.

- Improves software efficiency and effectiveness by simplifying how developers implement new features, functionalities, and integrations.

Why is it Used?

- APIs are crucial for creating integrated ecosystems where applications can leverage and extend each other’s capabilities in seamless, scalable ways.
- Enables developers to build upon existing platforms without having to start from scratch. This accelerates development cycles and promotes technology innovation.
- Facilitates secure and controlled access to service providers’ functionalities and data, enhancing the value and utility of software applications.
- Allows users to customize their interactions with various digital products and services, ensuring a tailored experience that meets specific needs.

XPERTS SUMMIT 2024

Types of APIs

Sub header

Types of APIs

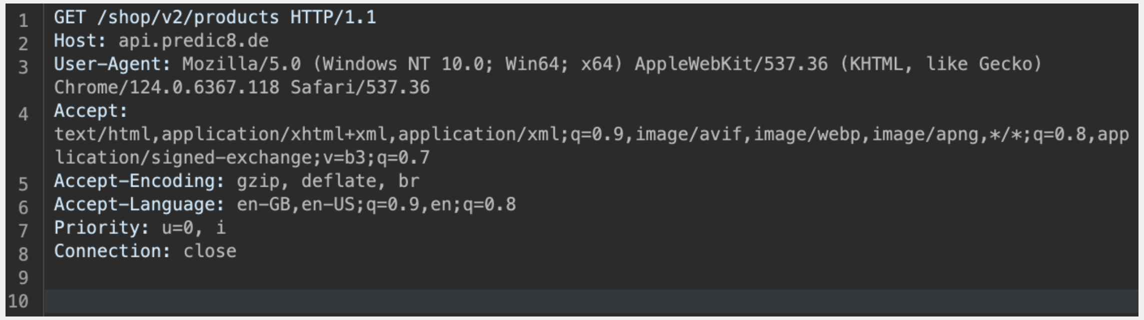
- REST APIs:
  - Architecture style using standard HTTP methods.
  - Stateless operations and scalable.
- gRPC:
  - Designed by Google, uses HTTP/2, suitable for internal communication.
  - Focus on performance; sends data as binary code.
- GraphQL:
  - Developed by Facebook; enables clients to request exactly the data they need.
  - No over or under-fetching issues.
- SOAP:
  - Standardized protocol with strict service interface.
  - Supports ACID compliance and transactionality.

REST based APIs

- REST (Representational State Transfer) is an architectural style for designing networked applications
- Best for public-facing services, straightforward CRUD (Create, Read, Update, Delete) applications.
- Principles:
  - Client-server architecture
  - Statelessness
  - Cacheability
  - Uniform interface
  - Layered system
- HTTP Methods used: GET, POST, PUT, DELETE, PATCH

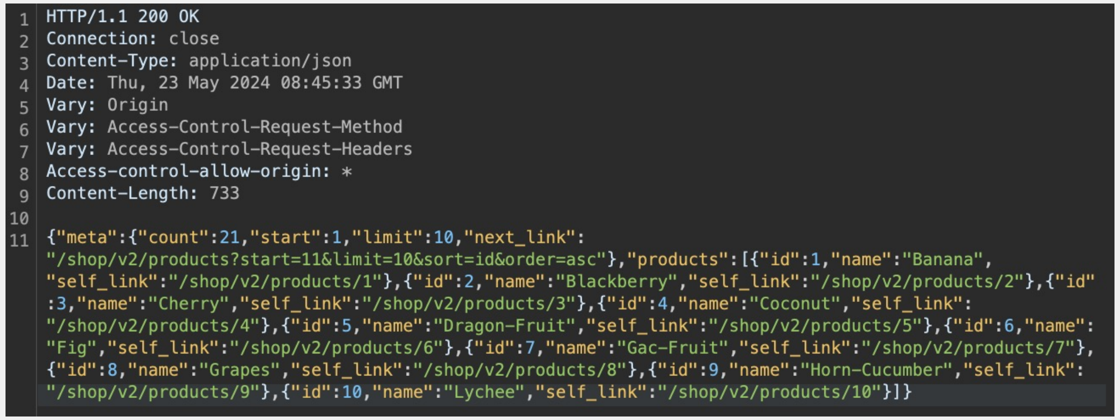
REST based APIs

HTTP Request



```
GET /shop/v2/products HTTP/1.1
Host: api.predic8.de
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/124.0.6367.118 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,app
lication/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=0, i
Connection: close


```

HTTP Response



```
HTTP/1.1 200 OK
Connection: close
Content-Type: application/json
Date: Thu, 23 May 2024 08:45:33 GMT
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-control-allow-origin: *
Content-Length: 733

{"meta":{"count":21,"start":1,"limit":10,"next_link":
"/shop/v2/products?start=11&limit=10&sort=id&order=asc"},"products":[{"id":1,"name":"Banana",
"self_link":"/shop/v2/products/1"},{"id":2,"name":"Blackberry","self_link":"/shop/v2/products/2"},{"id"
:3,"name":"Cherry","self_link":"/shop/v2/products/3"},{"id":4,"name":"Coconut","self_link":
"/shop/v2/products/4"},{"id":5,"name":"Dragon-Fruit","self_link":"/shop/v2/products/5"},{"id":6,"name":
"Fig","self_link":"/shop/v2/products/6"},{"id":7,"name":"Gac-Fruit","self_link":"/shop/v2/products/7"},
{"id":8,"name":"Grapes","self_link":"/shop/v2/products/8"},{"id":9,"name":"Horn-Cucumber","self_link":
"/shop/v2/products/9"},{"id":10,"name":"Lychee","self_link":"/shop/v2/products/10"}]}
```

gRPC based APIs

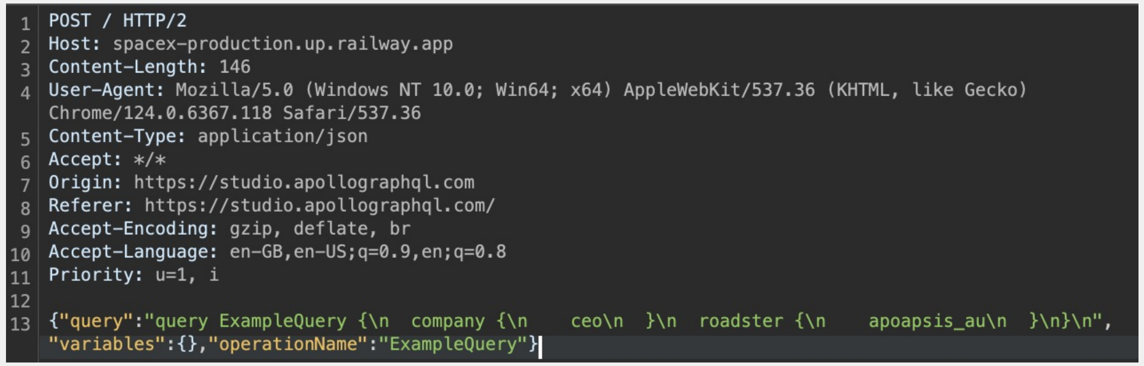
- gRPC (gRPC Remote Procedure Calls)
- developed by Google, underlying protocol is HTTP/2
- Best for microservices, high-performance APIs where low latency and high throughput are required.
- Features:
  - Starts with defining a service using Protobuf; specifies the methods that can be called remotely with their parameters and return types
  - Transmits data as binary
  - Supports server streaming, client streaming, and bidirectional streaming

GraphQL based APIs

- Developed by Facebook, GraphQL is a data query and manipulation language for APIs, and a runtime for fulfilling those queries.
- Useful for complex systems and applications where several resources need to be fetched in a single call.
- Minimizes over-fetching and under-fetching of data, easy integration with modern UI frameworks and libraries.
- Core Concepts:
  - Clients specify exact data requirements.
  - One endpoint: queries are expressive and return only what's necessary.

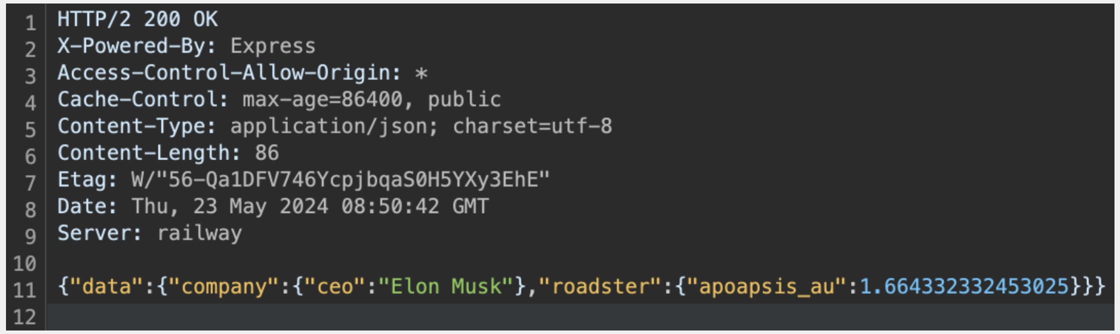
GraphQL based APIs

HTTP Request



```
POST / HTTP/2
Host: spacex-production.up.railway.app
Content-Length: 146
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/124.0.6367.118 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://studio.apollographql.com
Referer: https://studio.apollographql.com/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=1, i

{"query":"query ExampleQuery {\n  company {\n    ceo\n  }\n  roadster {\n    apoapsis_au\n  }\n}\n",
"variables":{},"operationName":"ExampleQuery"}
```

HTTP Response



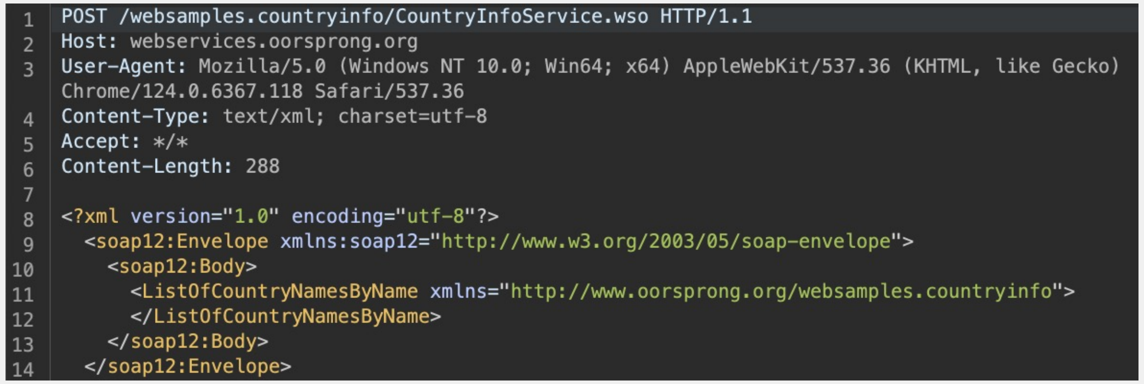
```
HTTP/2 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Cache-Control: max-age=86400, public
Content-Type: application/json; charset=utf-8
Content-Length: 86
Etag: W/"56-Qa1DFV746YcpjbqaS0H5YXy3EhE"
Date: Thu, 23 May 2024 08:50:42 GMT
Server: railway

{"data":{"company":{"ceo":"Elon Musk"},"roadster":{"apoapsis_au":1.664332332453025}}}
```

SOAP based APIs

- Simple Object Access Protocol (SOAP)
- Protocol specification for exchanging structured information in web services.
- Often used in enterprise environments where formal contracts, long-term compatibility, and enhanced security are required.
- Very secure, defined standards that everyone has to follow, suitable for distributed enterprise environments.
- Properties:
  - Protocol is based on XML to encode communication.
  - Follows a strict specification
  - must adhere to strict rules which allows for extended features like WS-Security, WS-Reliable Messaging
  - Description of API commonly stored within a WSDL file

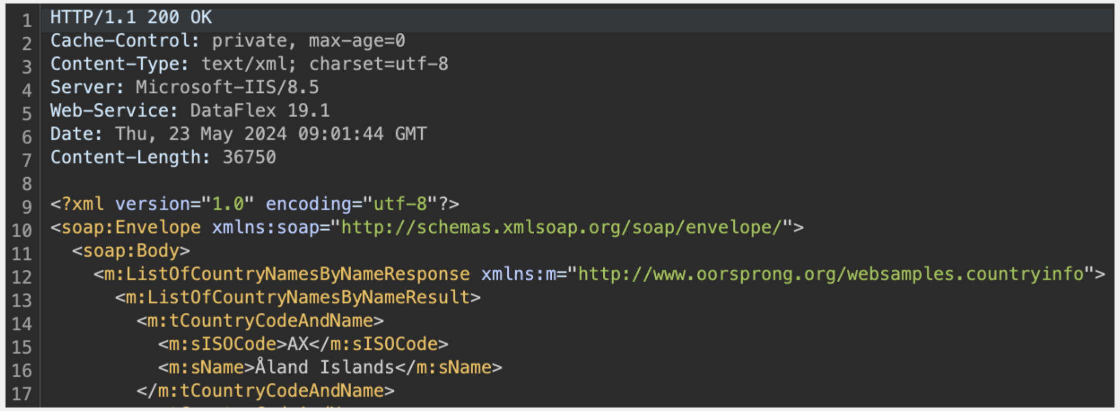
SOAP based APIs

HTTP Request



```
POST /websamples.countryinfo/CountryInfoService.wso HTTP/1.1
Host: webservices.oorsprong.org
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
Content-Type: text/xml; charset=utf-8
Accept: */*
Content-Length: 288

<?xml version="1.0" encoding="utf-8"?>
  <soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
    <soap12:Body>
      <ListOfCountryNamesByName xmlns="http://www.oorsprong.org/websamples.countryinfo">
      </ListOfCountryNamesByName>
    </soap12:Body>
  </soap12:Envelope>
```

HTTP Response



```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/8.5
Web-Service: DataFlex 19.1
Date: Thu, 23 May 2024 09:01:44 GMT
Content-Length: 36750

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <m:ListOfCountryNamesByNameResponse xmlns:m="http://www.oorsprong.org/websamples.countryinfo">
      <m:ListOfCountryNamesByNameResult>
        <m:tCountryCodeAndName>
          <m:sISOCode>AX</m:sISOCode>
          <m:sName>Åland Islands</m:sName>
        </m:tCountryCodeAndName>
```

Comparison of API Types

	Best	Good	Acceptable	Less Suiteable
Performance	gRPC	GraphQL	REST	SOAP
Flexibility	GraphQL	REST	SOAP	gRPC
Scalability	REST	gRPC	GraphQL	SOAP
Ease of Use	REST	GraphQL	SOAP	gRPC
Security	SOAP	REST	gRPC	GraphQL

XPERTS SUMMIT 2024

API Authentication

Sub header

API Authentication

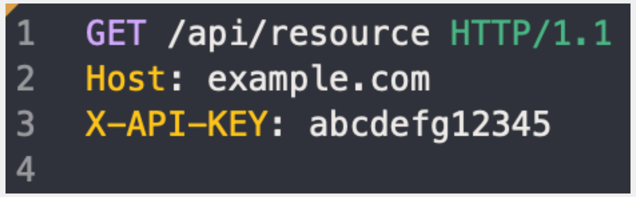
- Mechanisms to verify identity for security.
- Common Methods:
  - API keys
  - OAuth
  - JSON Web Tokens (JWT)

Key based Authentication (API Keys)

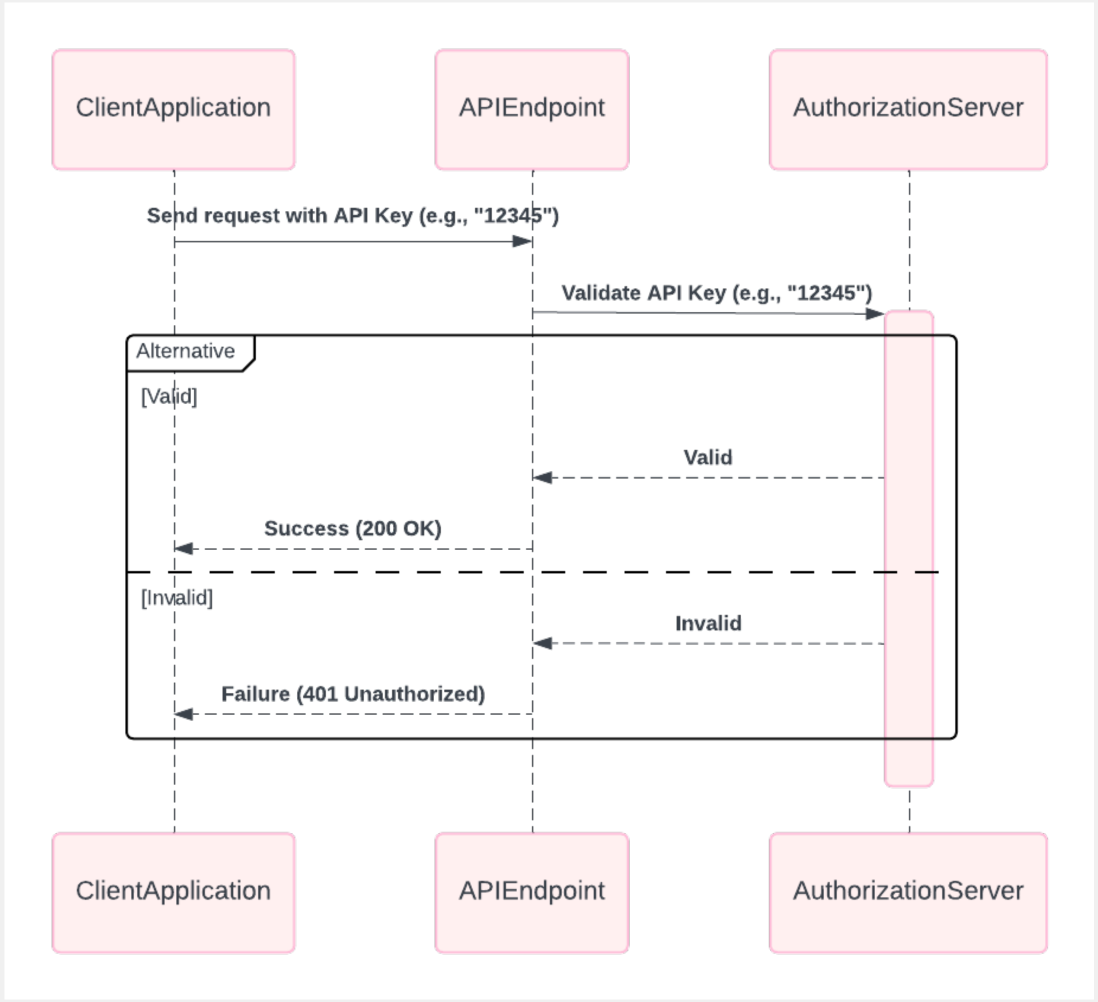
- API Keys as a simple, reproducible token
- passed along with API requests
- Simple to implement and use, immediate access control.
- Often used in client-server interactions where the risk level is low to moderate.
- Example:
  - Web services where extensive permissions and user management are not required.

Key based Authentication (API Keys)

- Location:
  - Query string
  - Request header
  - Cookie
- Example Request:



```
GET /api/resource HTTP/1.1
Host: example.com
X-API-KEY: abcdefg12345

```

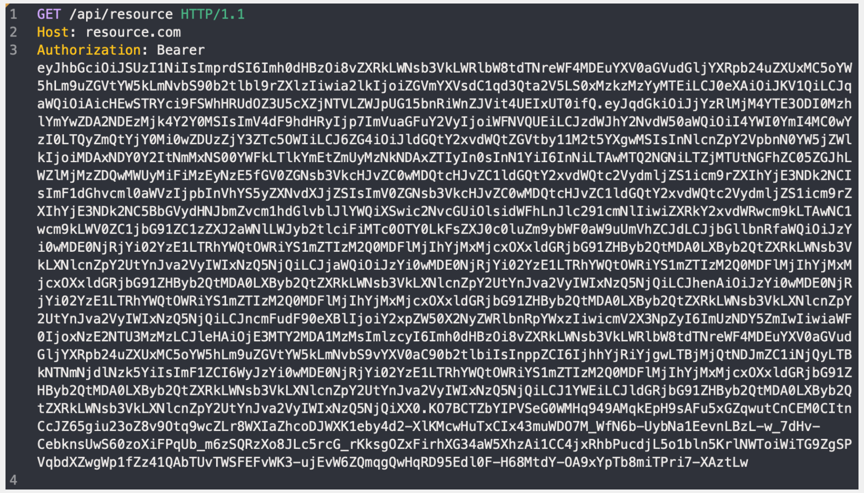


OAuth

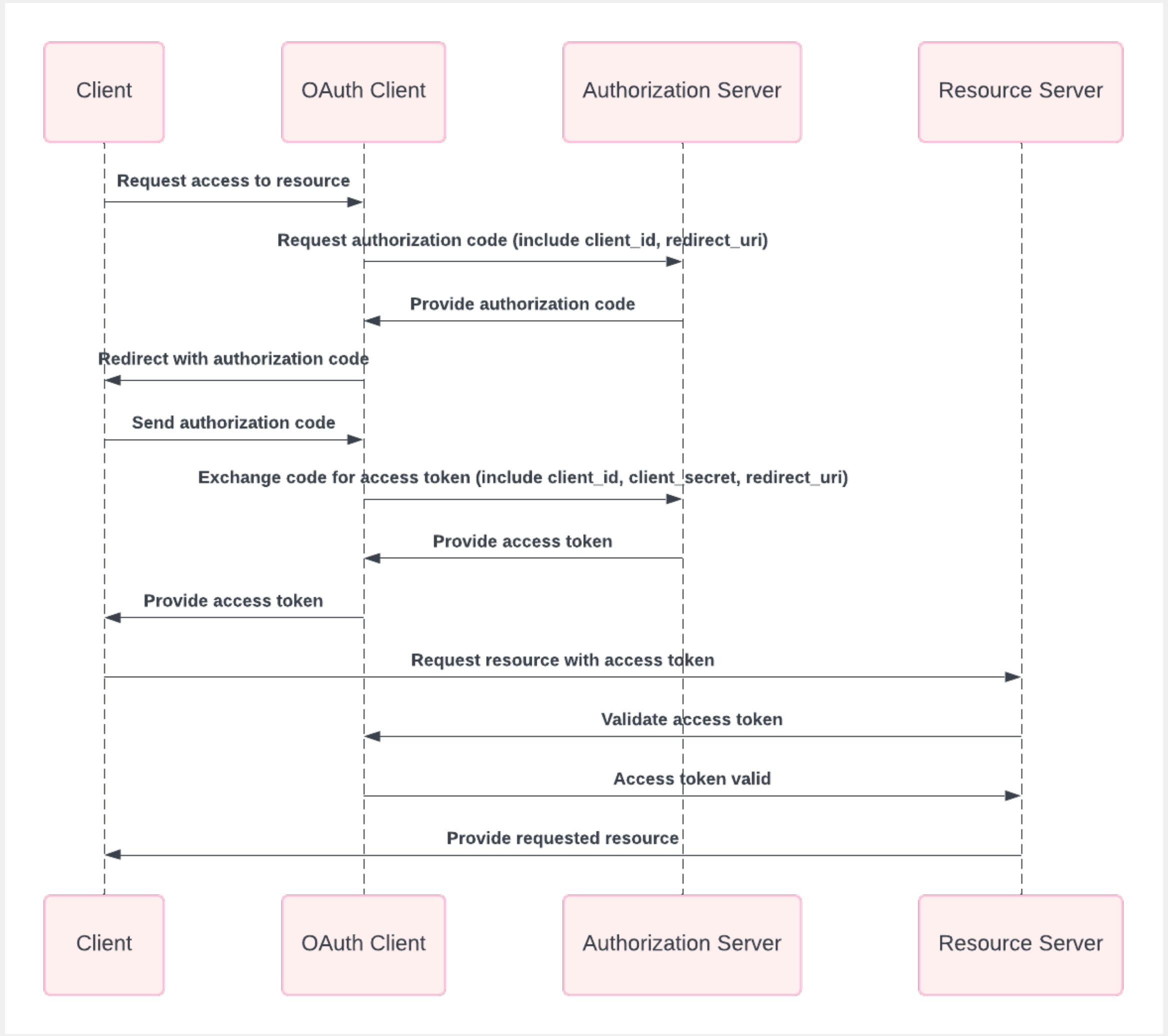
- a robust authorization framework
- Not authentication – only authorization
- allowing third-party services to exchange web resources without exposing credentials
- requiring complex and tailored access permissions
- Ideal for services where users need to access resources from multiple systems without compromising on security.

OAuth

- Location:
  - Authorization header
- Example:



```
GET /api/resource HTTP/1.1
Host: resource.com
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImprdSI6Imh0dHBzOi8vZXRkLWNsb3VkLWRlbW8tdTNreWF4MDEuYXV0aGVudGljYXRpb24uZXUxMC5oYW
5hLm9uZGVtYW5kLmNvbS90b2tlbl9rZXlzIiwia2lkIjoiZGVmYXVsdC1qd3Qta2V5LS0xMzkzMzYyMTEiLCJ0eXAiOiJKV1QiLCJq
aWQiOiAicHEwSTRYci9FSWhHRUdOZ3U5cXZjNTVLZWJpUG15bnRiWnZJVit4UEIxUT0ifQ.eyJqdGkiOiJjYzRlMjM4YTE3ODI0Mzh
lYmYwZDA2NDEzMjk4Y2Y0MSIsImV4dF9hdHRyIjp7ImVuaGFuY2VyIjoiWFNVQUEiLCJzdWJhY2NvdW50aWQiOiI4YWI0YmI4MC0wY
zI0LTQyZmQtYjY0Mi0wZDUzZjY3ZTc5OWIiLCJ6ZG4iOiJldGQtY2xvdWQtZGVtby11M2t5YXgwMSIsInNlcnZpY2VpbnN0YW5jZWl
kIjoiMDAxNDY0Y2ItNmMxNS00YWFkLTlkYmEtZmUyMzNkNDAxZTIyIn0sInN1YiI6InNiLTAwMTQ2NGNiLTZjMTUtNGFhZC05ZGJhL
WZlMjMzZDQwMWUyMiFiMzEyNzE5fGV0ZGNsb3VkcHJvZC0wMDQtcHJvZC1ldGQtY2xvdWQtc2VydmljZS1icm9rZXIhYjE3NDk2NCI
sImF1dGhvcml0aWVzIjpbInVhYS5yZXNvdXJjZSIsImV0ZGNsb3VkcHJvZC0wMDQtcHJvZC1ldGQtY2xvdWQtc2VydmljZS1icm9rZ
XIhYjE3NDk2NC5BbGVydHNJbmZvcm1hdGlvblJlYWQiXSwic2NvcGUiOlsidWFhLnJlc291cmNlIiwiZXRkY2xvdWRwcm9kLTAwNC1
wcm9kLWV0ZC1jbG91ZC1zZXJ2aWNlLWJyb2tlciFiMTc0OTY0LkFsZXJ0c0luZm9ybWF0aW9uUmVhZCJdLCJjbGllbnRfaWQiOiJzY
i0wMDE0NjRjYi02YzE1LTRhYWQtOWRiYS1mZTIzM2Q0MDFlMjIhYjMxMjcxOXxldGRjbG91ZHByb2QtMDA0LXByb2QtZXRkLWNsb3V
kLXNlcnZpY2UtYnJva2VyIWIxNzQ5NjQiLCJjaWQiOiJzYi0wMDE0NjRjYi02YzE1LTRhYWQtOWRiYS1mZTIzM2Q0MDFlMjIhYjMxM
jcxOXxldGRjbG91ZHByb2QtMDA0LXByb2QtZXRkLWNsb3VkLXNlcnZpY2UtYnJva2VyIWIxNzQ5NjQiLCJhenAiOiJzYi0wMDE0NjR
jYi02YzE1LTRhYWQtOWRiYS1mZTIzM2Q0MDFlMjIhYjMxMjcxOXxldGRjbG91ZHByb2QtMDA0LXByb2QtZXRkLWNsb3VkLXNlcnZpY
2UtYnJva2VyIWIxNzQ5NjQiLCJncmFudF90eXBlIjoiY2xpZW50X2NyZWRlbnRpYWxzIiwicmV2X3NpZyI6ImUzNDY5ZmIwIiwiaWF
0IjoxNzE2NTU3MzMzLCJleHAiOjE3MTY2MDA1MzMsImlzcyI6Imh0dHBzOi8vZXRkLWNsb3VkLWRlbW8tdTNreWF4MDEuYXV0aGVud
GljYXRpb24uZXUxMC5oYW5hLm9uZGVtYW5kLmNvbS9vYXV0aC90b2tlbiIsInppZCI6IjhhYjRiYjgwLTBjMjQtNDJmZC1iNjQyLTB
kNTNmNjdlNzk5YiIsImF1ZCI6WyJzYi0wMDE0NjRjYi02YzE1LTRhYWQtOWRiYS1mZTIzM2Q0MDFlMjIhYjMxMjcxOXxldGRjbG91Z
HByb2QtMDA0LXByb2QtZXRkLWNsb3VkLXNlcnZpY2UtYnJva2VyIWIxNzQ5NjQiLCJ1YWEiLCJldGRjbG91ZHByb2QtMDA0LXByb2Q
tZXRkLWNsb3VkLXNlcnZpY2UtYnJva2VyIWIxNzQ5NjQiXX0.KO7BCTZbYIPVSeG0WMHq949AMqkEpH9sAFu5xGZqwutCnCEM0CItn
CcJZ65giu23oZ8v9Otq9wcZLr8WXIaZhcoDJWXK1eby4d2-XlKMcwHuTxCIx43muWDO7M_WfN6b-UybNa1EevnLBzL-w_7dHv-
CebknsUwS60zoXiFPqUb_m6zSQRzXo8JLc5rcG_rKksgOZxFirhXG34aW5XhzAi1CC4jxRhbPucdjL5o1bln5KrlNWToiWiTG9ZgSP
VqbdXZwgWp1fZz41QAbTUvTWSFEFvWK3-ujEvW6ZQmqgQwHqRD95Edl0F-H68MtdY-OA9xYpTb8miTPri7-XAztLw
```


Client
OAuth Client
Authorization Server
Resource Server
Request access to resource
Request authorization code (include client_id, redirect_uri)
Provide authorization code
Redirect with authorization code
Send authorization code
Exchange code for access token (include client_id, client_secret, redirect_uri)
Provide access token
Provide access token
Request resource with access token
Validate access token
Access token valid
Provide requested resource
Client
OAuth Client
Authorization Server
Resource Server

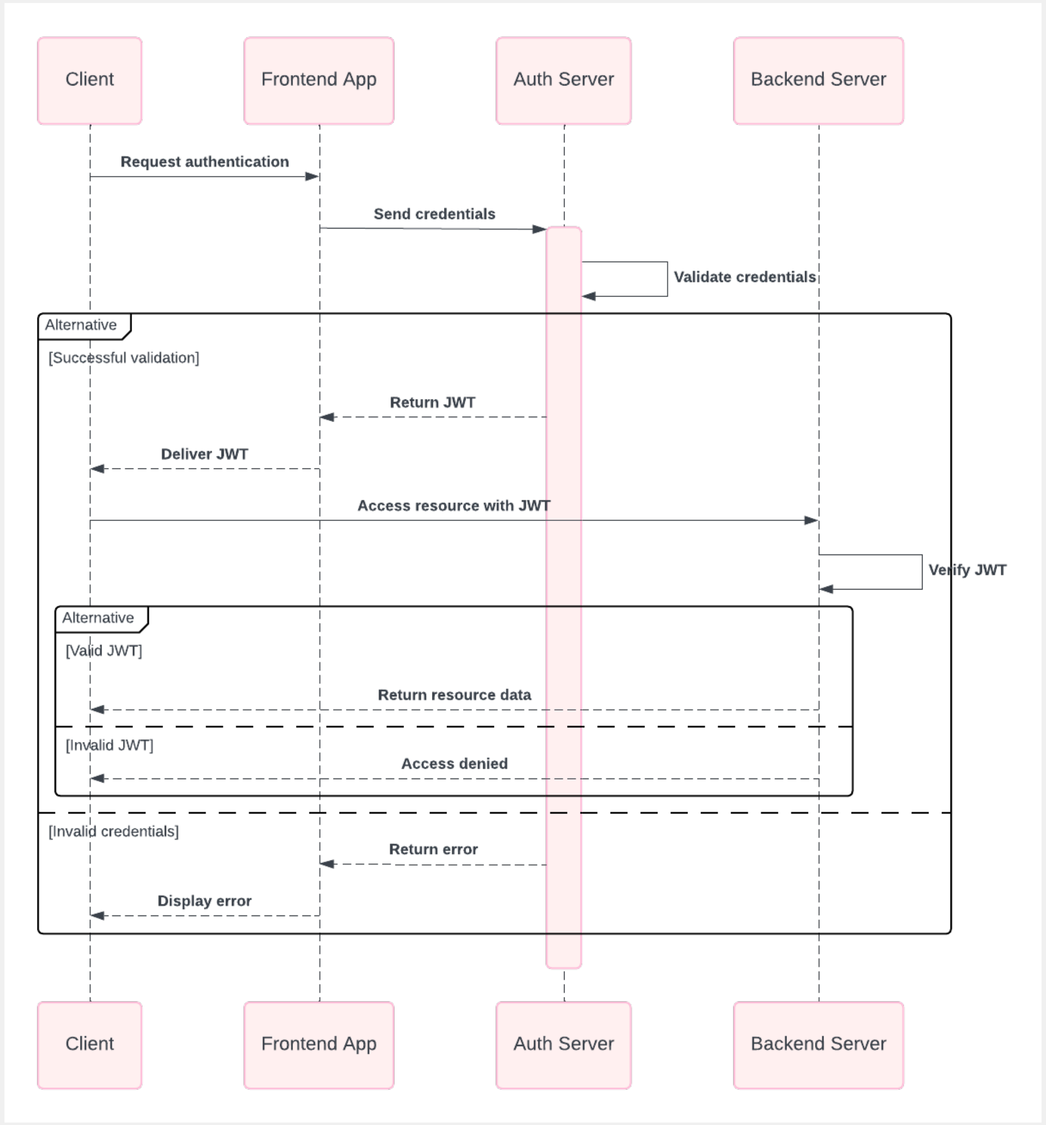
JSON Web Token (JWT)

- securely transmitting information between parties as a JSON object, signed and URL-safe
- can assert claims
  - Who issued the token
  - How long is it valid
  - What permissions do the client have
- Location:
  - Authorization header

JSON Web Token (JWT)

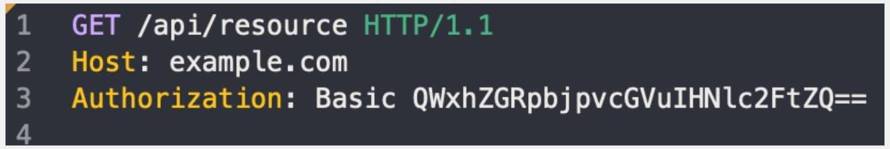
- Components of JWT:
  - Header
    - Algorithm
    - type
  - Payload
    - Claims / data
  - Signature
    - Signature to verify integrity of data
- Example:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

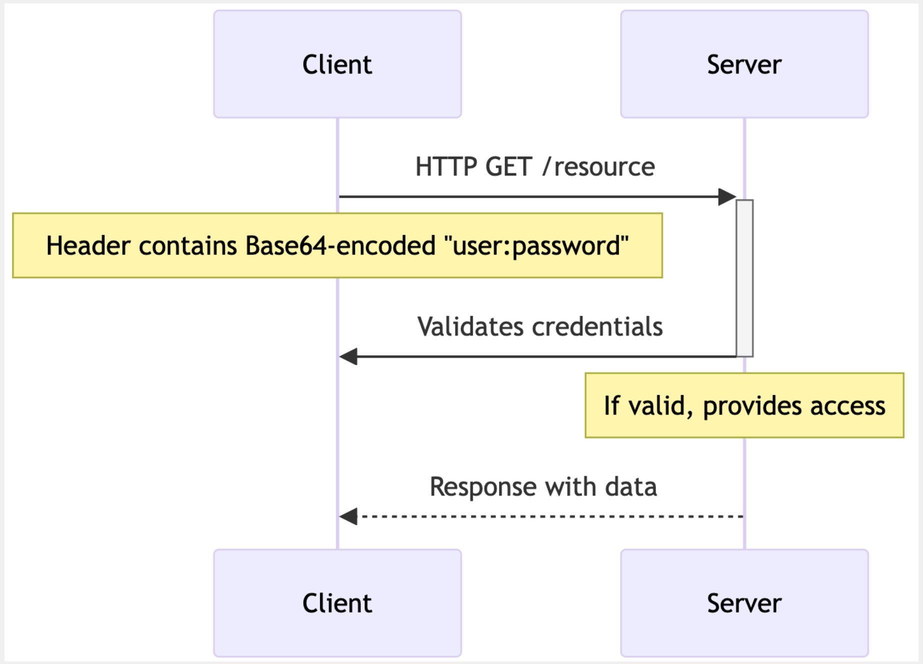


Basic Authentication

- user credentials (username and password) are encoded and sent with HTTP requests
- Vulnerable to eavesdropping
- Good for simple internal applications
- Example:



```
GET /api/resource HTTP/1.1
Host: example.com
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

```



OWASP API Top 10

Sub header

Overview

- API1:2023 - Broken Object Level Authorization
- API2:2023 - Broken Authentication
- API3:2023 - Broken Object Property Level Authorization
- API4:2023 - Unrestricted Resource Consumption
- API5:2023 - Broken Function Level Authorization
- API6:2023 - Unrestricted Access to Sensitive Business Flows
- API7:2023 - Server Side Request Forgery
- API8:2023 - Security Misconfiguration
- API9:2023 - Improper Inventory Management
- API10:2023 - Unsafe Consumption of APIs

API1: Broken Object Level Authorization

- **Risk Description**

Insufficient checks which allow attackers to manipulate object properties to access unauthorized data.

- **Attack Example**

An attacker modifies the *userId* parameter in an API call to access data belonging to another user.

- **Explanation**

Often occurs in systems where ACLs (Access Control Lists) and RBAC (Role-Based Access Control) are not properly enforced or incorrectly implemented.

API2: Broken Authentication

- **Risk Description**

API security features related to authentication are improperly implemented, allowing attackers to compromise authentication tokens or exploit implementation flaws.

- **Attack Example**

An attacker uses stolen authentication tokens to gain unauthorized access to the system.

- **Explanation**

This risk is prevalent when the API does not securely handle the session tokens, especially in multi-step processes like multi-factor authentication.

API3: Broken Object Property Level Authorization

- **Risk Description**

APIs improperly expose sensitive object properties, often due to inadequate data filtering based on user roles.

- **Attack Example**

An API endpoint returns user details including passwordHash or isAdmin due to insufficient data serialization controls.

- **Explanation**

This vulnerability allows unauthorized users to access or modify sensitive properties, potentially leading to elevated privileges.

API4: Unrestricted Resource Consumption

- **Risk Description**

Occurs when APIs fail to limit the use of resources, leading to potential denial of service or degraded performance due to excessive consumption of bandwidth, memory, CPU, or disk space.

- **Attack Example**

An attacker sends an extraordinary number of requests or huge payloads in a short span, overwhelming the API server resources.

- **Explanation**

Unrestricted resource consumption can render services slow or entirely unavailable, affecting all users.

API5: Broken Function Level Authorization

- **Risk Description**

Different API endpoints implement function level authorization differently, leading to unauthorized access.

- **Attack Example**

An attacker accesses an admin only function through manipulating parameters or the access route.

- **Explanation**

This happens when permissions are improperly coded at different layers or are inconsistent across an application.

API6: Unrestricted Access to Sensitive Business Flows

- **Risk Description**

This risk occurs when sensitive business processes are not properly protected, allowing unauthorized users to access or manipulate them, potentially leading to loss of data or service integrity.

- **Attack Example**

An attacker exploits weak access controls to initiate a high-value financial transaction without proper authorization.

- **Explanation**

Exposure of sensitive business flows can lead to significant financial, reputational, or compliance risks.

API7: Server Side Request Forgery (SSRF)

- **Risk Description**

SSRF occurs when an attacker abuses a server functionality to make requests to internal resources, or to external ones, that the server should not have access to. This can lead to information disclosure, internal system access, or acting as a pivoting point in the network.

- **Attack Example**

An attacker exploits a vulnerable API that fetches a URL from a user input without validation. They manipulate it to cause the server to make a call to internal services like databases or administrative interfaces.

- **Explanation**

SSRF can expose internal services and sensitive data, potentially allowing attackers to leverage the server to perform malicious requests.

API8: Security Misconfiguration

- **Risk Description**

Security misconfiguration happens when APIs are not securely set up or are configured using defaults. This can lead to unnecessary vulnerabilities due to unsecured services, overly informative error messages, or improperly managed credentials.

- **Attack Example**

An attacker exploits default administrative credentials left on a database exposed by an API, allowing unauthorized data access.

- **Explanation**

Common configurations issues include verbose error messages revealing stack traces, unnecessary features and services running, or open cloud storage.

API9: Improper Inventory Management

- **Risk Description**

Improper Inventory Management refers to the oversight or mismanagement of tracking and securing API endpoints and versions. This risk arises when organizations lack a comprehensive overview of their API assets, leading to unidentified, outdated, or unsupervised APIs which may involve serious security risks.

- **Attack Example**

An attacker discovers an obsolete version of an API still in operation but not properly secured or updated, exploiting known vulnerabilities that allow unauthorized access to sensitive data.

- **Explanation**

Inadequate API inventory management can lead to unmonitored endpoints, outdated API versions, and APIs with deprecated functions being accessible, all of which increase the security risk.

API10:Unsafe Consumption of APIs

- **Risk Description**

Unsafe Consumption of APIs refers to the hazardous use or integration of external APIs without adequate security checks, validations, or oversight. This can lead to vulnerabilities in an application due to trusting external sources too implicitly.

- **Attack Example**

An application integrates an external API that includes malicious code or behaves unexpectedly under certain conditions, leading to data leakage or loss.

- **Explanation**

Relying on APIs without fully understanding their security posture or without implementing robust error handling can expose applications to various risks such as data corruption, unauthorized data exposure, and denial of service.

XPERTS SUMMIT 2024

Tools for API Development and Testing

Sub header

Command line Tools

- curl (Linux/Unix/MacOS)
- httpie
- hrpcurl / grpc_cli

GUI Applications

- Postman
- BurpSuite/ZAP/Caido
- SoapUI
- GraphQL Playground