Full Stack Application Development

Securing applications

Akshaya Ganesan

Agenda

- Basic Authentication
- API Keys
- JWT
- OAuth

Security

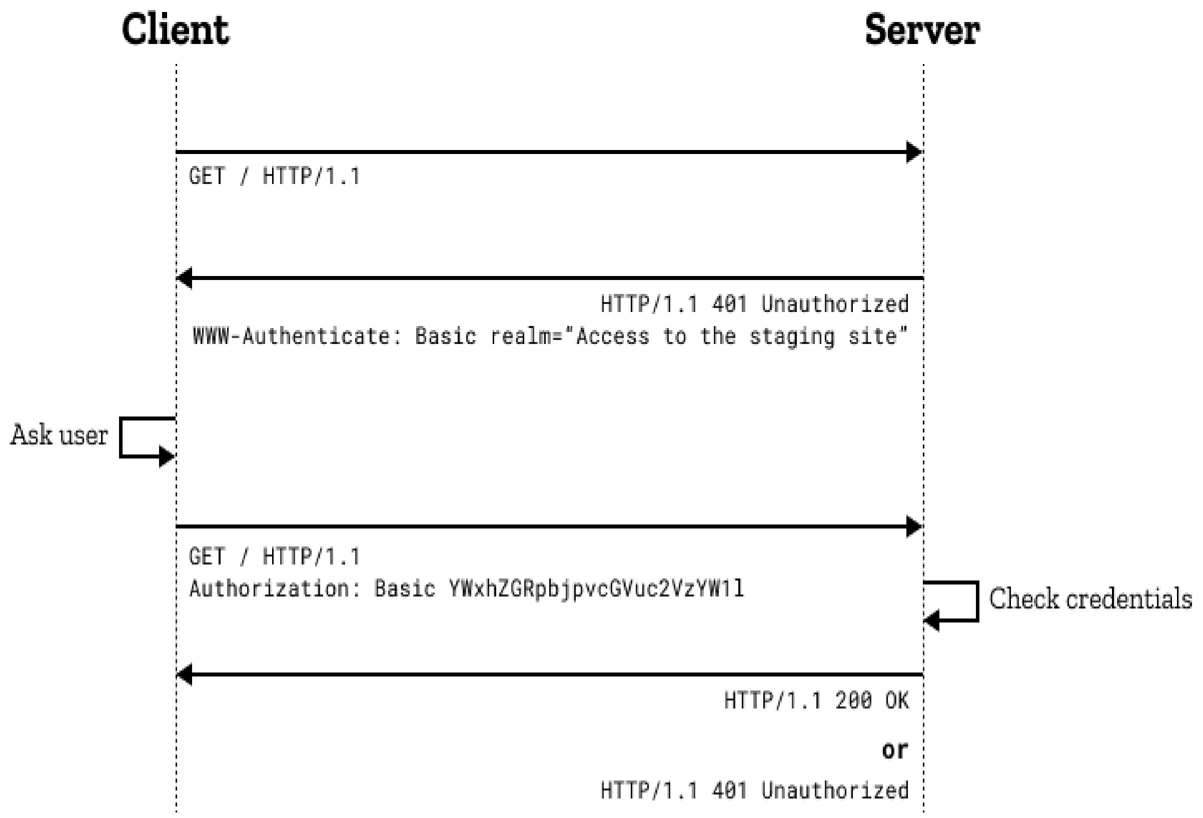
- Authentication and authorization are two foundation elements of security:
- Authentication is the process of verifying who a user is.
- Authorization is the process of verifying what they have access to.

Basic Authentication

- HTTP provides a general framework for access control and authentication.
- In basic HTTP authentication, a request contains a header field in the form of

  Authorization: Basic <credentials>

- The credentials is the Base64 encoding of username and password joined by a single colon
- Basic authentication is typically used in conjunction with HTTPS to provide confidentiality.



Client
Server
GET / HTTP/1.1
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="Access to the staging site"
Ask user
GET / HTTP/1.1
Authorization: Basic YWxhZGRpbjpvcGVuc2VzYW1l
Check credentials
HTTP/1.1 200 OK
or
HTTP/1.1 401 Unauthorized

Image Ref:https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

Basic Authentication

- Here are some reasons why it is still used:
    - Simplicity
    - Compatibility
    - Statelessness
- Limitations
    - Security
    - Single Factor
    - Risk of Credential Exposure

API Keys

- An **API key** is a unique identifier used to authenticate and authorize the access to an **API**
- Instead of a username and password, the client application is issued a unique API key, typically a long alphanumeric string.
- The API key is sent in the HTTP request as a parameter in the query string or the request headers
- (e.g., api_key=your_api_key or Authorization: API-Key your_api_key).
- API keys are commonly used for machine-to-machine communication or applications interacting with the API.
- Some APIs use API keys to enforce rate limits.

API Keys

- **Security**: Treat API keys as sensitive information. Avoid hardcoding them in client-side code or exposing them publicly.
- **Rotation**: Regularly rotate API keys to enhance security. You can invalidate a key and issue a new one if a key is compromised.
- **Scopes**: Consider using different keys for different purposes (e.g., read-only vs. administrative access).
- **HTTPS**: Always use HTTPS to transmit API keys securely.
- Examples: **Google Maps API, Cloud Services**:

Token-based authentication system

- Token-based authentication allows users to verify their identity, and in return receive a unique access token.
- Token-based authentication is different from traditional password-based technique.
- In stateless communication, each request that the user makes to the server contains all the necessary information for authentication, typically in the form of token.
- The server validates the token and responds accordingly for each request.

Types Of Tokens

- **Opaque tokens**
- The opaque token is a random, unique string of characters the authorization server issues.
- The opaque token does not pass any identifiable information
- To validate the token and retrieve the information on the token and the user, the resource server calls the authorization server and requests the token introspection.
- **Structured token:**
- Its format is well-defined so the resource server can decode and verify the token without calling the authorization server.
- JWT is a structured token

JSON Web Tokens (JWT)

- JSON Web Tokens (JWTs) are a format of tokens used in web development and security.
- JWT is a standard way to securely represent claims, such as user identity and roles, between two parties.
- A JWT has a payload, which is a JSON object that contains information about the user, such as their identity and roles, and other metadata, such as an expiration date.
- It’s signed with a secret that’s only known to the creator of the JWT.
- The secret ensures a malicious third party can’t forge or tamper with a JWT.

JSON Web Tokens (JWT)

- JSON Web Tokens consist of three parts separated by dots (.), which are:
  - Header
  - Payload
  - Signature
- Therefore, a JWT typically looks like the following: xxxxx.yyyyy.zzzzz

JSON Web Tokens (JWT)

- **Header**
- The header typically consists of two parts: the type of the token, which is JWT, and the hashing algorithm such as

  HMAC SHA256 or RSA.
- Then, this JSON is Base64Url encoded to form the first part of the JWT.
- **Payload**
- The second part of the token is the payload, which contains the claims.
- Claims are statements about an entity (typically, the user) and additional metadata.
- The payload is then **Base64Url** encoded to form the second part of the JWT.

Header:
```
{
"alg": "HS256",
  "typ": "JWT"
}
```

Payload:
```
{
   "sub": "1234567890",
   "name": "John Doe",
    "admin": true
}
```

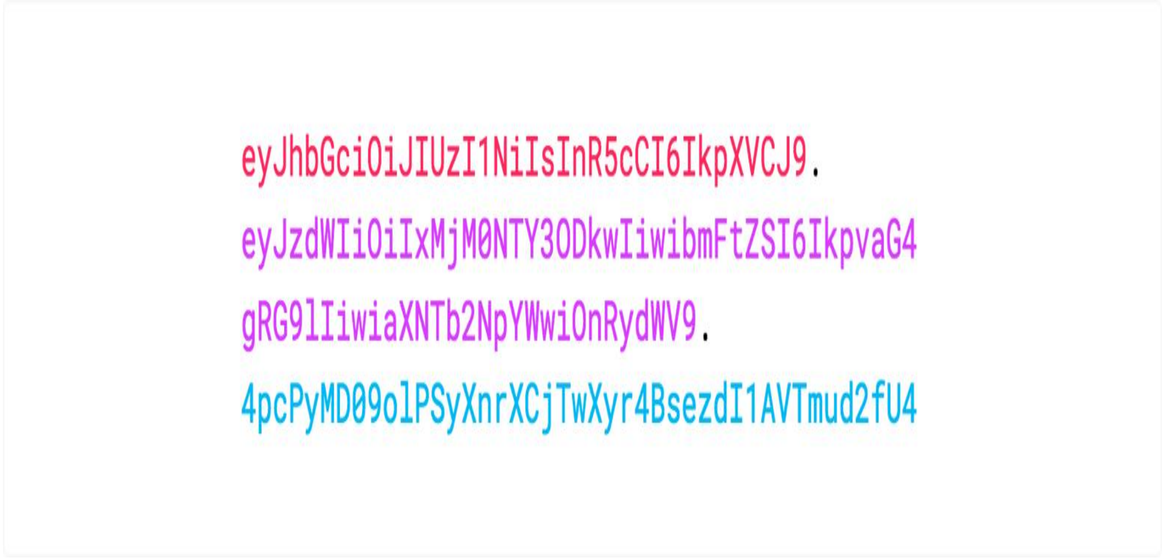
JSON Web Tokens (JWT)

- **Signature**

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

```
HMACSHA256(
  base64UrlEncode(header) + '.' +
  base64UrlEncode(payload),
  secret)
```

- The signature is used to verify that the sender of the JWT
- The output is three Base64 strings separated by dots



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

Uses of JWT

- Information Exchange
- Single Sign on
- Authentication and Authorization

Thank You!