infogain

Spring JWT Security

Objectives

- Introduction to JWT
- When should you use JSON Web Tokens?
- What is the JSON Web Token structure?
- JWT Authentication Flow

What is JSON Web Token?

- **What is JSON Web Token?**
  - JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.
  - This information can be verified and trusted because it is digitally signed.
  - JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA.

- **Let's explain some concepts of this definition further.**
  - **Compact:** Because of their smaller size, JWTs can be sent through a URL, POST parameter, or inside an HTTP header. Additionally, the smaller size means transmission is fast.
  - **Self-contained:** The payload contains all the required information about the user, avoiding the need to query the database more than once.

When should you use JSON Web Tokens?

- **Here are some scenarios where JSON Web Tokens are useful:**

  - **Authorization:** This is the most common scenario for using JWT. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used across different domains.
  - **Information Exchange:** JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed—for example, using public/private key pairs—you can be sure the senders are who they say they are. Additionally, as the signature is calculated using the header

What is the JSON Web Token structure?

- **What is the JSON Web Token structure?**
- 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 Token structure(contd..)

- **Let's break down the different parts.**
- **Header**
- The header typically consists of two parts: the type of the token, which is JWT, and the hashing algorithm being used, such as HMAC SHA256 or RSA.
- **For example:**

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

- Then, this JSON is Base64Url encoded to form the first part of the JWT.

JSON Web Token structure(contd..)

- **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.
- **There are three types of claims: reserved, public, and private claims.**
  - **Reserved claims:** These are a set of predefined claims which are not mandatory but recommended, to provide a set of useful, interoperable claims. Some of them are: iss (issuer), exp (expiration time), sub (subject) and others.
  - **Notice that the claim names are only three characters long as JWT is meant to be compact.**
  - **Private claims:** These are the custom claims created to share information between parties that agree on using them.

- **An example of payload could be:**

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

- The payload is then Base64Url encoded to form the second part of the JSON Web Token.

JSON Web Token structure(contd..)

- **Signature**
  - The signature is used to verify that the sender of the JWT is who it says it is and to ensure that the message wasn't changed along the way.
  - 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.

- **For example if you want to use the HMAC SHA256 algorithm, the signature will be created in the following way:**

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

JSON Web Token structure(contd..)

- **Putting all together**
- The output is three Base64 strings separated by dots that can be easily passed in HTML and HTTP environments, while being more compact when compared to XML-based standards such as SAML.
- **The following shows a JWT that has the previous header and payload encoded, and it is signed with a secret. Encoded JWT**

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

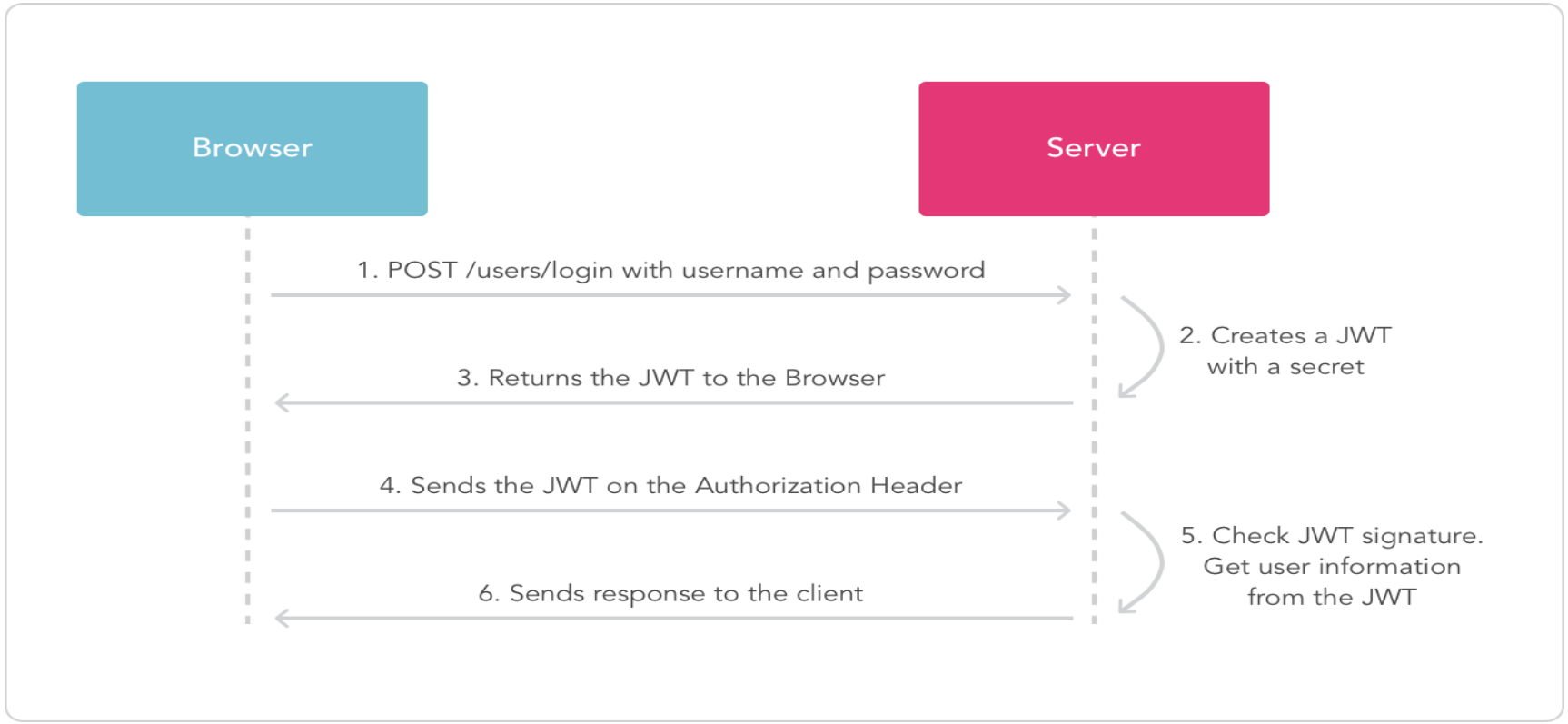
How do JSON Web Tokens work?

- **How do JSON Web Tokens work?**
- In authentication, when the user successfully logs in using their credentials, a JSON Web Token will be returned.
- Whenever the user wants to access a protected route or resource, the user agent should send the JWT, typically in the Authorization header using the Bearer schema. The content of the header should look like the following:
- **Authorization: Bearer <token>**

- This is a stateless authentication mechanism as the user state is never saved in server memory. The server's protected routes will check for a valid JWT in the Authorization header, and if it's present, the user will be allowed to access protected resources. As JWTs are self-contained, all the necessary information is there, reducing the need to query the database multiple times.

How do JSON Web Tokens work(contd..)

- **The following diagram shows this process:**



infogain

Thank You

infogain

www.infogain.com

Infogain Corporation, HQ
485 Alberto Way Los Gatos,
CA 95032 USA
Phone: 408-355-6000
Fax: 408-355-7000

Pune
7th Floor, Bhalerao Towers, CTS No.1669 - 1670, Behind Hotel Pride,
Shivaji Nagar, Pune - 411005
Phone : +91-20-66236700

Infogain Irvine
41 Corporate Park,
Suite 390 Irvine, CA 2606 USA
Phone: 949-223-5100
Fax: 949-223-5110

Infogain Austin
Stratum Executive Center Building D
11044 Research Boulevard Suite 200
Austin, Texas 78759

Noida
A-16, Sector 60, Noida Gautam Budh agar,
201301 (U.P.) India
Phone: +91-120-2445144
Fax: +91-120-2580406

Dubai
P O Box 500588 Office No.105,
Building No. 4, Dubai Outsource Zone,
Dubai, United Arab Emirates
Tel: +971-4-458-7336