

Advanced API Security

ITANA Group

Nuwan Dias

Architect

22/06/2017

Agenda

- Introduction to API Security
- WSO2 API Security Architecture
- API Authentication
- API Authorization
- Fine Grained Access Control
- Providing Javascript Apps secure access to APIs
- Securing B2B APIs using OAuth2.0
- Data Redaction

HTTP Basic Authentication

- **Creating a GitHub repository**

```
curl -l
```

```
-u $GitHubUserName:$GitHubPassword
```

```
-X POST -H 'Content-Type: application/x-www-form-urlencoded'
```

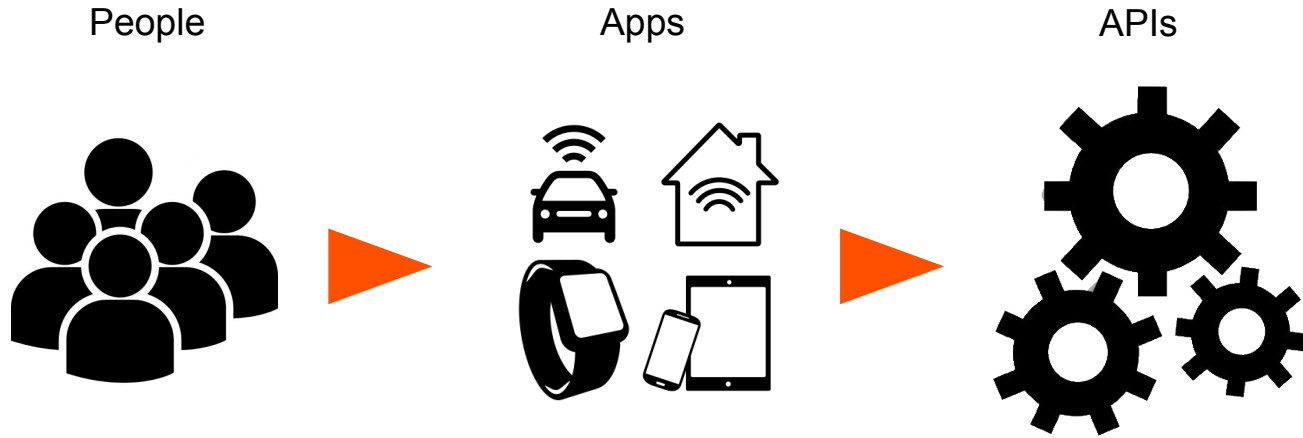
```
-d '{"name": "my_github_repo"}'
```

```
https://api.github.com/user/repos
```

```
Authorization: Basic QWxhZGRpbjppPcGVuU2VzYW1l
```

API Security is about controlling Access

Delegation



OAuth2.0

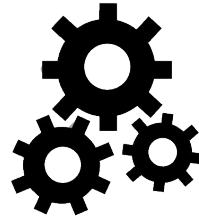
- A Framework that has mastered the art of Access Delegation.
- Depends on SSL/TLS.
- Caters a wide variety of use cases via Grant Types.



Resource
Owner



Client
Application



Resource
Server



Authorization
Server

OAuth2.0

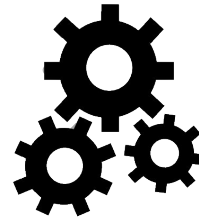
- A Framework that has mastered the art of Access Delegation.
- Depends on SSL/TLS.
- Caters a wide variety of use cases via Grant Types.



Resource
Owner



Client
Application



Resource
Server



Authorization
Server



OAuth2.0

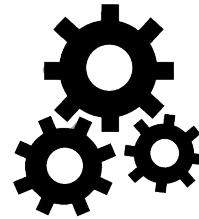
- A Framework that has mastered the art of Access Delegation.
- Depends on SSL/TLS.
- Caters a wide variety of use cases via Grant Types.



Resource
Owner



Client
Application



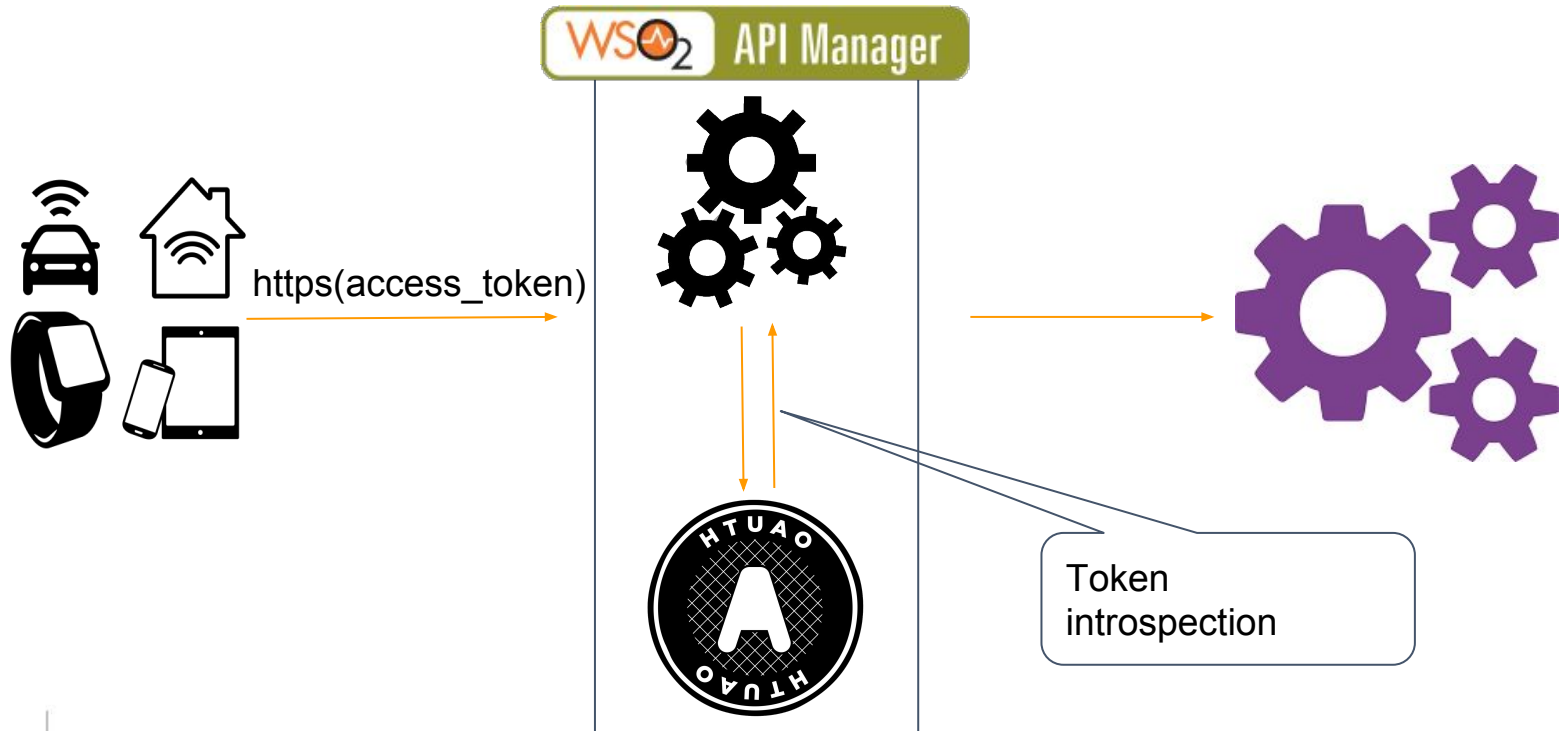
API Gateway



Key Manager



WSO2 API Security Architecture



API Authentication - Getting an Access Token

- Access Token Request

```
curl -k -d "grant_type=password&username=<username>&password=<password>" -H  
"Authorization :Basic base64encode(consumer-key:consumer-secret), Content-Type:  
application/x-www-form-urlencoded" https://localhost:8243/token
```

- Access Token Response

```
{"token_type":"bearer","expires_in":3600,  
"refresh_token":"8579facb65d1d3eba74a395a2e78dd6",  
"access_token":"eb51eff0b4d85cda1eb1d312c5b6a3b8"}
```

API Authorization - Scopes

- A scope defines an action to be performed.
- A scope can be bound to one or many roles/groups.
- A scope can be attached to one or more API Resources.
- Scopes are granted to an access tokens.
- Scopes have to be requested for when requesting for the access token.

API Authorization - Scopes

Define Scope ×

Scope Key

Scope Name

Roles

Description

API Authorization - Scopes

Resources

Scopes

-  news_read : Read News
Roles : employee, manager
-  news_write : Post News
Roles : manager

 Add Scopes

/read

GET

/read + Summary

Application & Application User

Unlimited

Read News

/write

POST

/write + Summary

Application & Application User

Unlimited

Post News

Getting an Access Token with Scope(s)

- Access Token Request

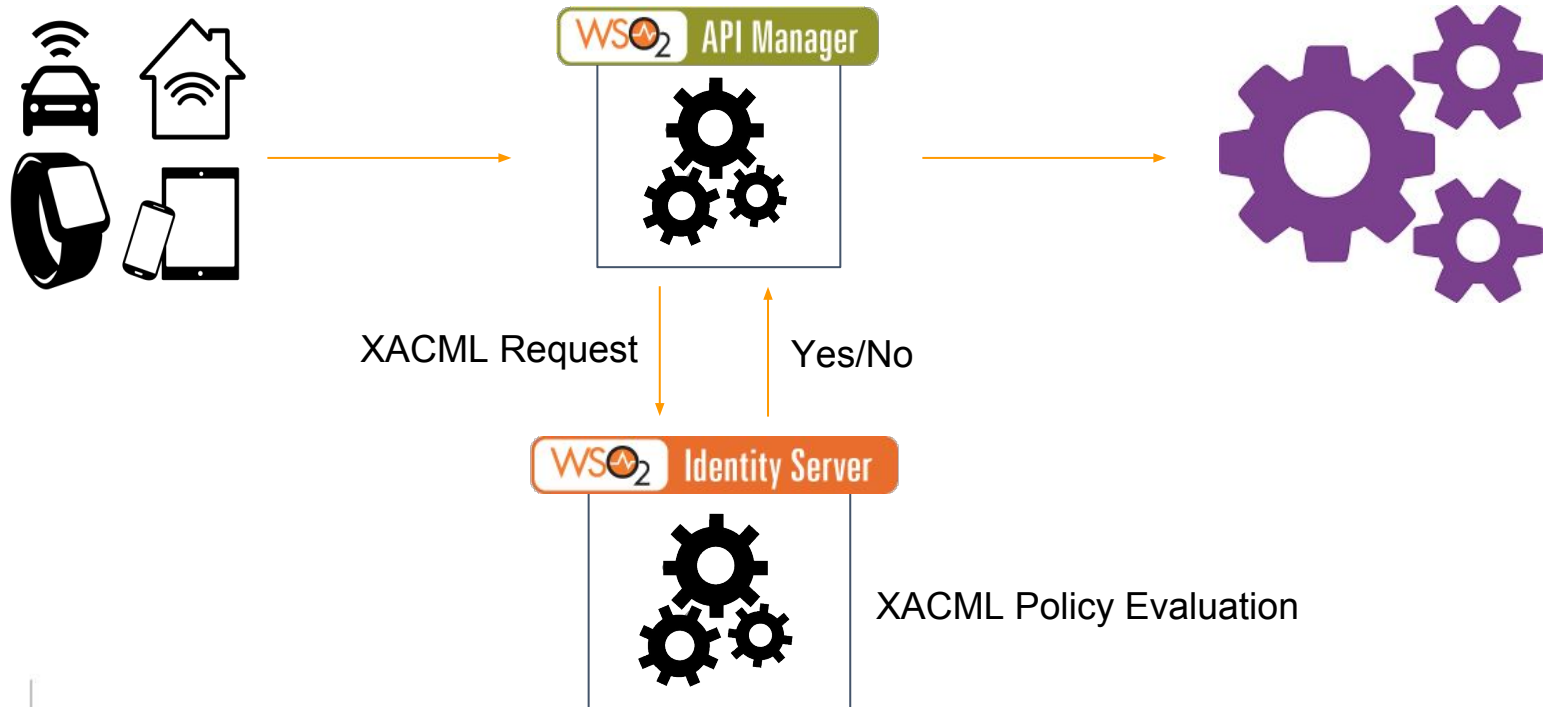
```
curl -k -d "grant_type=password&username=<username>&password=<password>&scope=news_read  
news_write" -H "Authorization :Basic base64encode(consumer-key:consumer-secret), Content-Type:  
application/x-www-form-urlencoded" https://localhost:8243/token
```

- Access Token Response

```
{"scope":"news_read", "token_type":"bearer","expires_in":3600,  
"refresh_token":"8579facb65d1d3eba74a395a2e78dd6",  
"access_token":"eb51eff0b4d85cda1eb1d312c5b6a3b8"}
```

Fine Grained Access Control using Policies

- XACML provides a way to perform policy based fine grained access control



XACML Reference Architecture

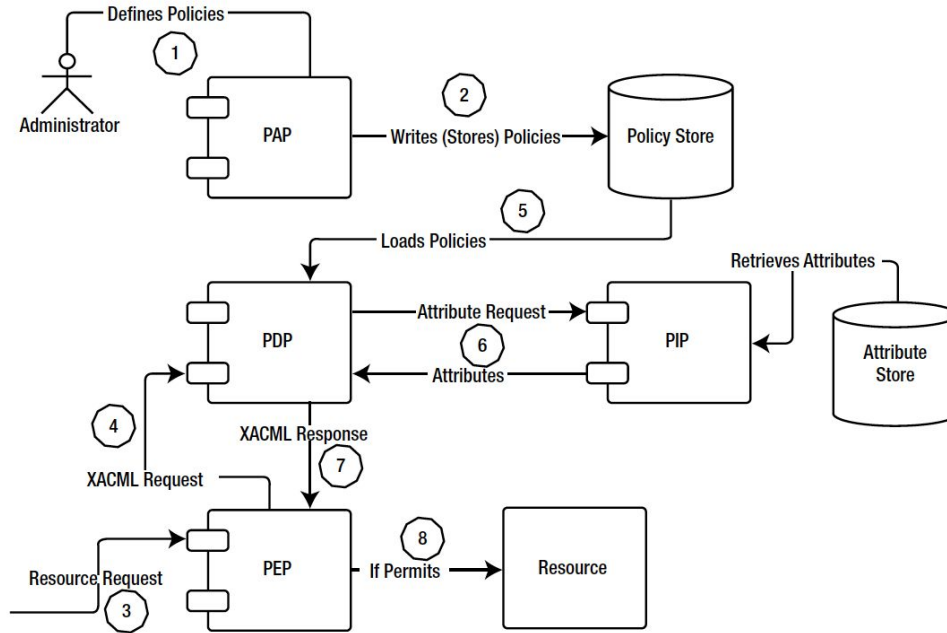


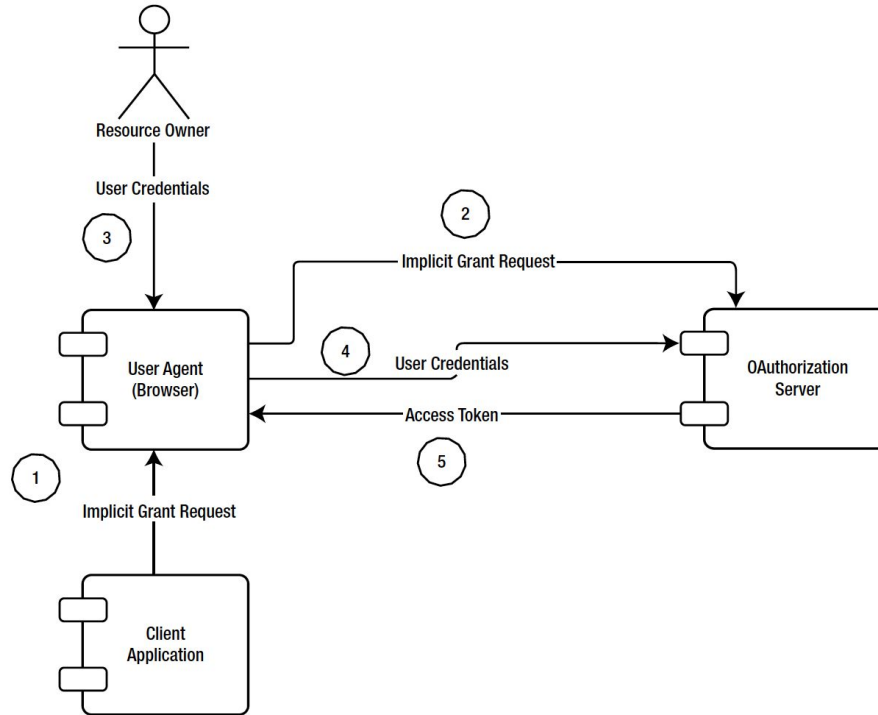
Image Credits: Advanced API Security by Prabath Siriwardena

OAuth2.0 - Implicit Grant

- The problem - Javascript Applications aren't capable of securely storing sensitive information.
- The solution - The implicit Grant protocol doesn't require the client application to store sensitive information nor expects the user to provide credentials to the client application.
- The token is sent in a redirect URL to the browser as a URI fragment. URI Fragments are not submitted to the server and only accessible by Javascript.

https://myexamplecallback.com/#access_token=asdfwe-asdab243-asn3sl&expires_in=3600

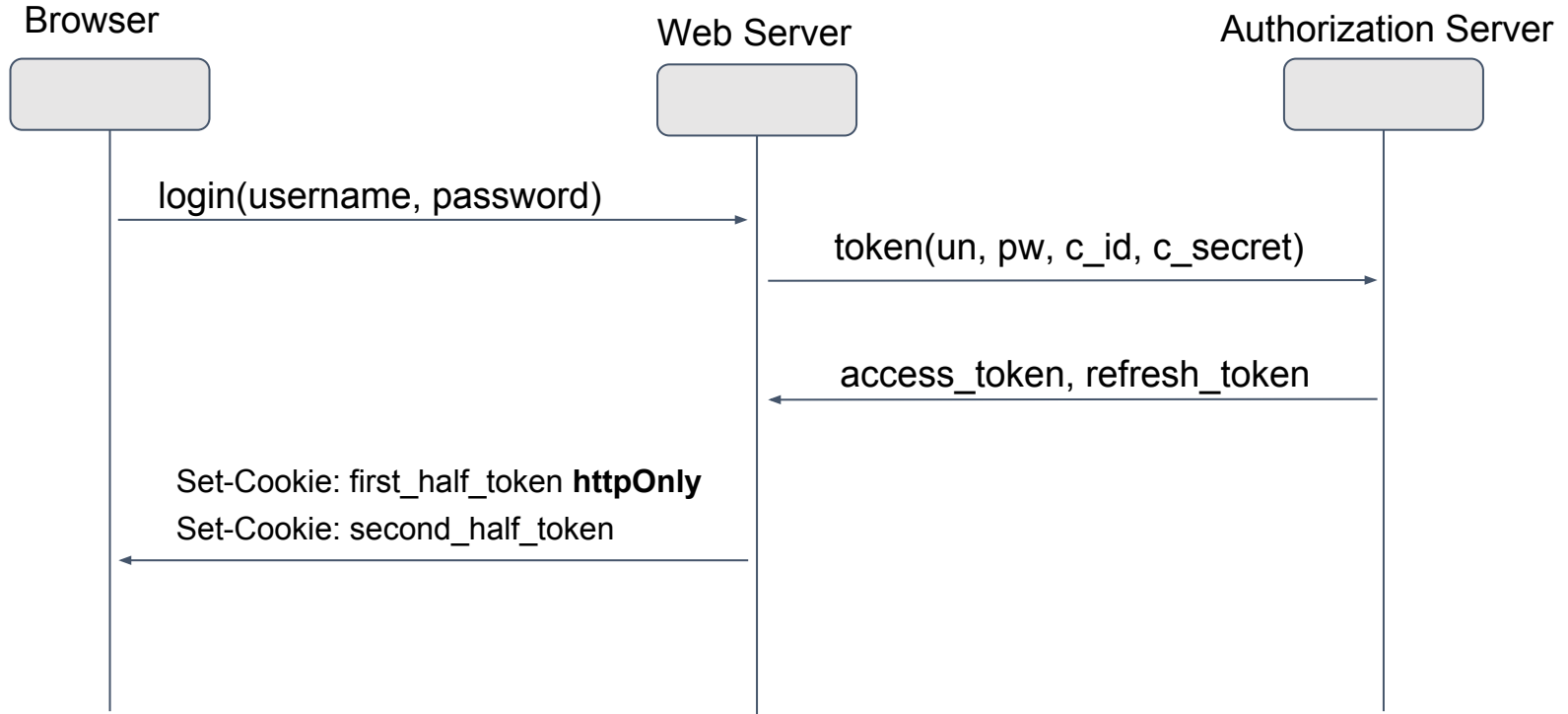
OAuth2.0 - Implicit Grant



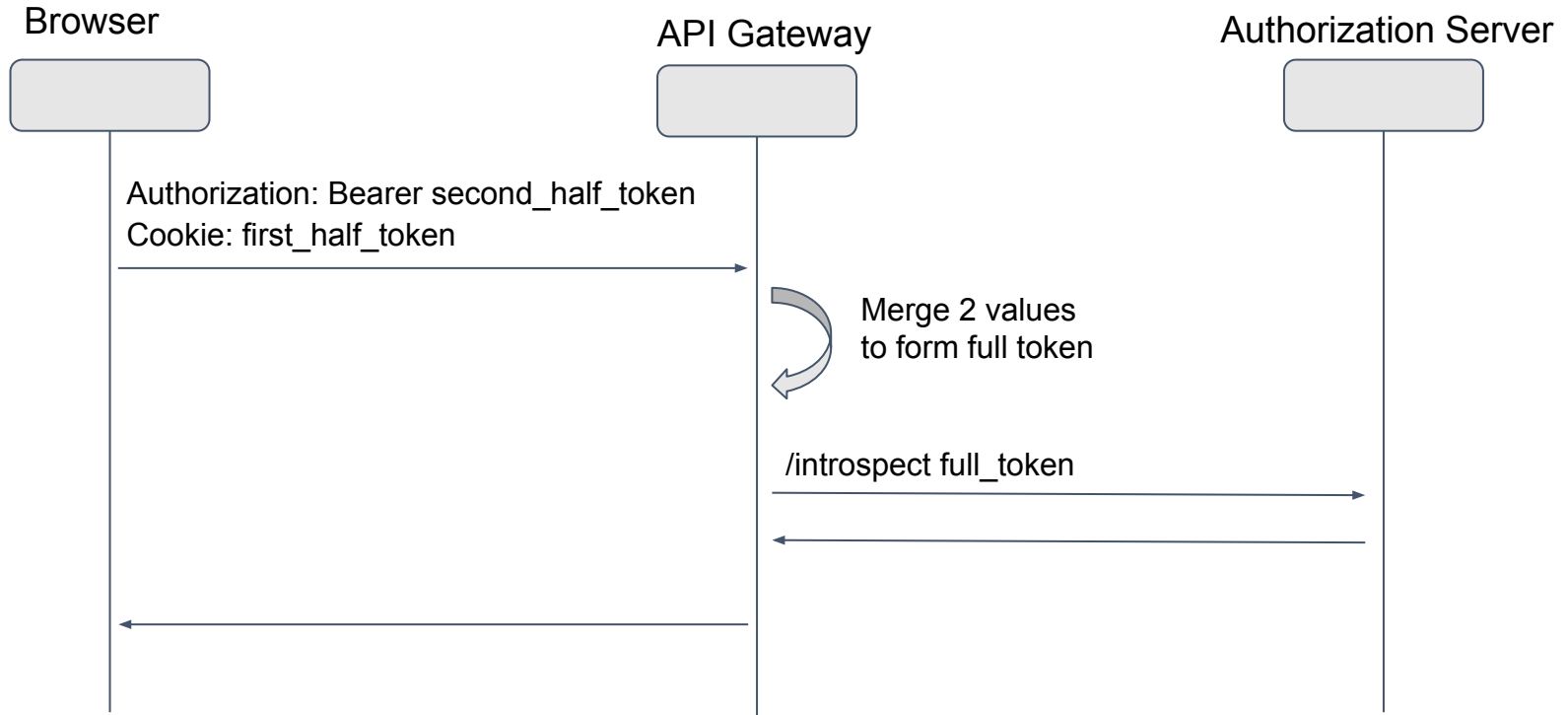
Few drawbacks with the Implicit Grant

- Does not have a refresh token - Forces users to re-login when token expires
- Vulnerable to XSS attacks - If the site is vulnerable to XSS, an attacker can steal the token

A possible alternative - The split token pattern



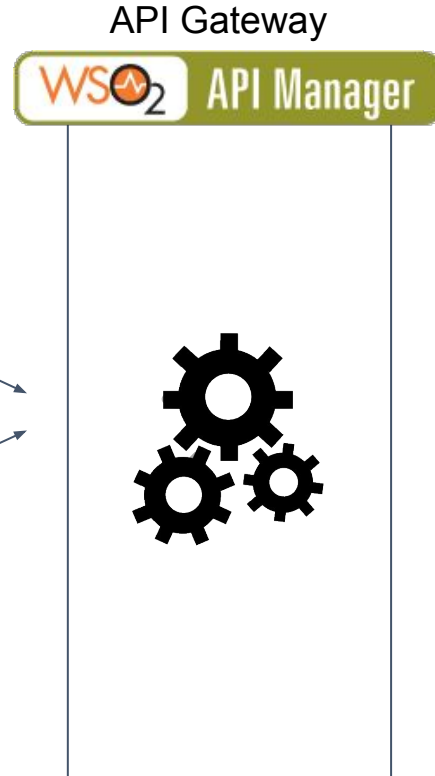
Split token pattern - Token Validation



Securing B2B APIs using OAuth

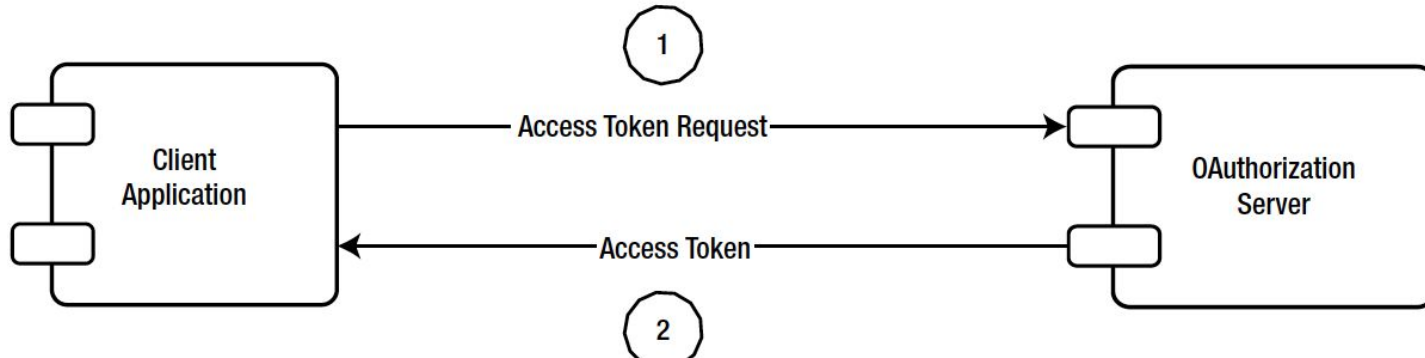
- The problem - B2B do not have a human interaction. This can result in complications when a user is required to authenticate himself to get a token.
- The solutions -
 - client_credentials grant - For unique client applications (no resource owner)
 - JWT grant - For reusable client applications with a resource owner

client_credentials grant

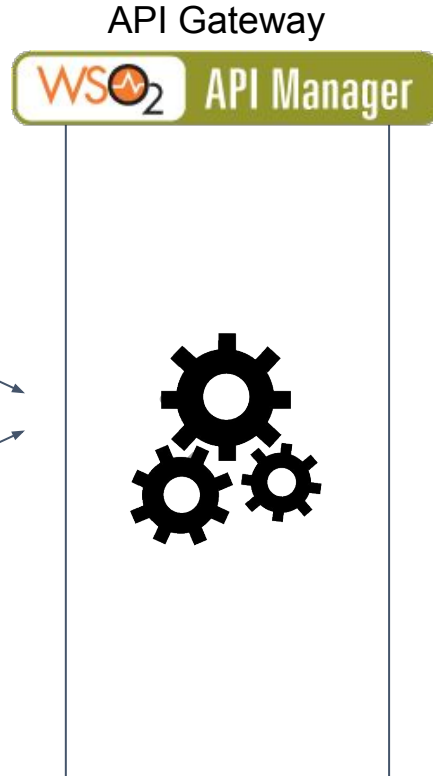
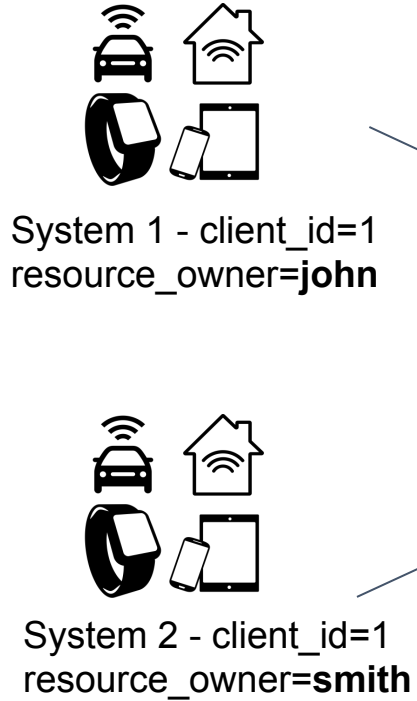


client_credentials grant

```
curl -v -X POST --basic  
-u 0rhQErxIX49svVYoXJGt0DWBuFca:eYOfkL756W8usQaVNgCNkz9C2D0a  
-H "Content-Type:application/x-www-form-urlencoded;  
charset=UTF-8" -k  
-d "grant_type=client_credentials" https://localhost:9443/oauth2/token
```



JWT Grant



JWT

eyJhbGciOiJIUzI1NiJ9.eyJleHAiOjE0NTUyYyR_UhKuPWNJw



JWT Header



JWT Payload



JWT Signature

Sample Header

```
{
  "alg": "RS256"
}
```

Sample Payload

```
{
  "exp": 1458166985,
  "sub": "john",
  "nbf": 1458106985,
  "aud": [
    "https://localhost:9443/oauth2/token",
    "wso2-IS"
  ],
  .....
}
```

JWT Grant

Client Application



Authorization Server



```
POST /token HTTP/1.1
Host: auth.bar.com
Content-Type: application/x-www-form-urlencoded
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer
&assertion=eewewbGciOiJFUzewqew.eewew3Mi[...omitted
for brevity...].
ewe-ZhwP[...omitted for brevity...]
```

1. Decode JWT
2. Validate JWT Signature
3. Extract 'sub'
4. Validate 'sub' against requested scopes
5. Issue access token to client.

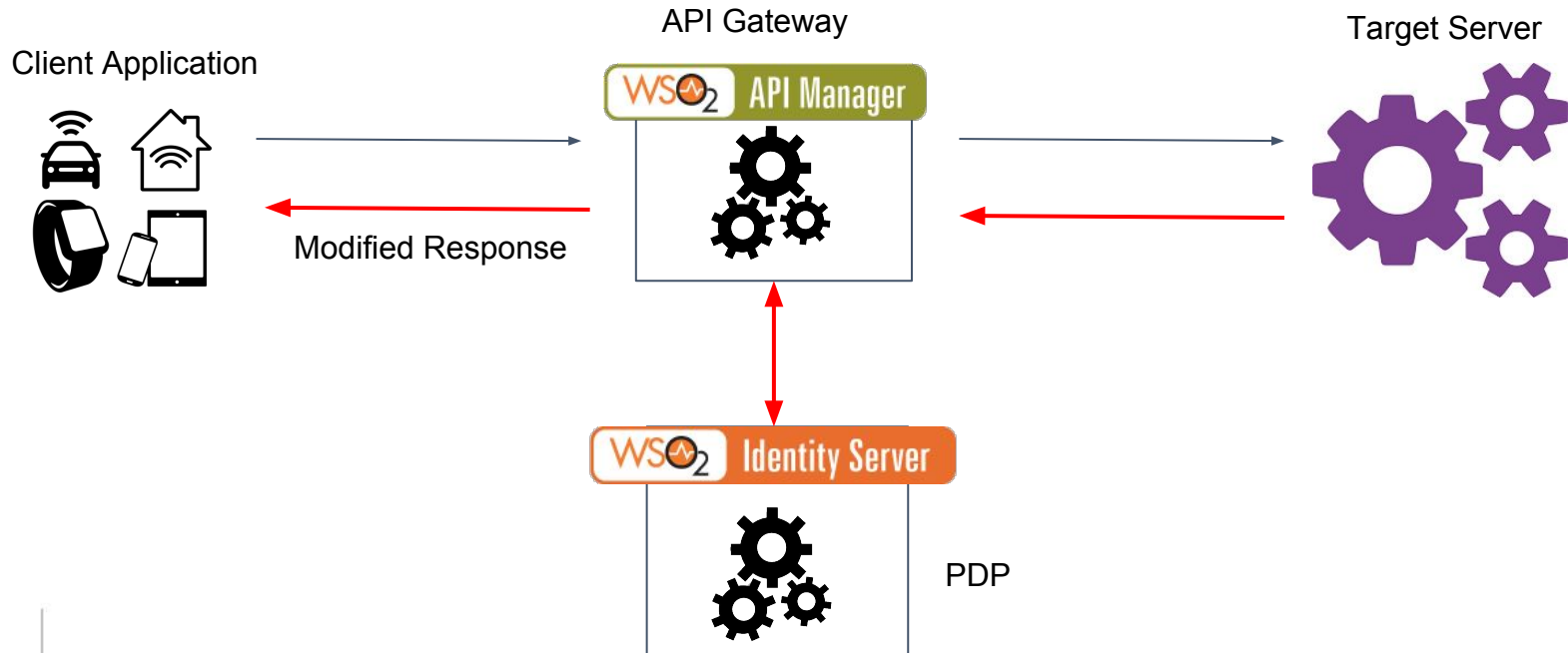
Data Redaction - Method 1 (Through the Target Server)

- The API Gateway generates a JWT that contains the user claims and other attributes and passes this information to the Target Server (back-end) in a special http header.
- The Target Server can use this information to decide which data to be provided in the response.

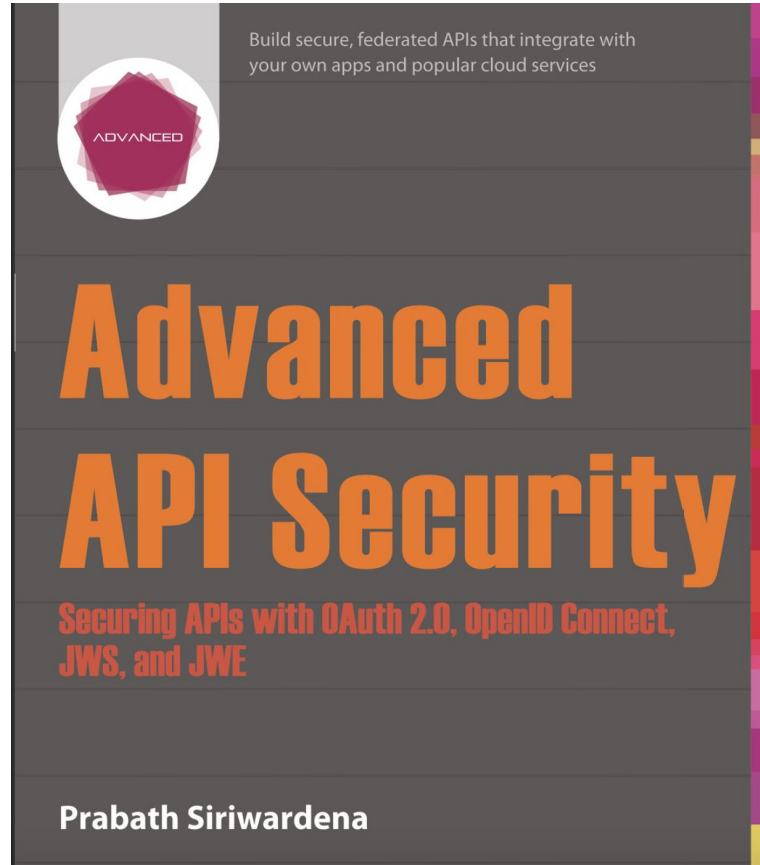


Data Redaction - Method 2 (Using Policies)

- The API Gateway contacts a PDP to check if the user bears a necessary permission and determine the final API response based on those facts



Recommended Reading



THANK YOU

wsO2.com

