API Security for APEX

govtech stack

Topics

API Security

- Introduction
- Use HTTPS
- Types of API Security Policy
- Utilities and Demo
- Discussion

apex

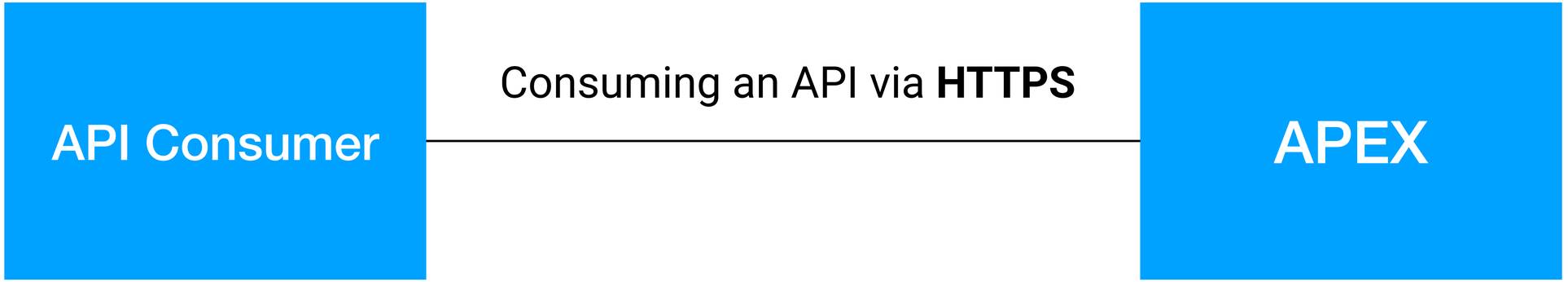
Robin Cher
https://github.com/robincher

Wei Junyu
https://github.com/weijunyu

Introduction - API Security

- Unauthorised Access
- Audit Requirements
- Why? **Protect and secure your assets**

Context



Today discussion is about how we secure this leg

1. Use HTTPS
2. Basic Auth
3. HTTP Signature
4. JOSE

Use HTTPS

- REST APIs should only be served securely through HTTPS
- Use TLS 1.1 and TLS 1.2 **(Recommended)**
- No more **SSL** Please!

Types of API Security Policy

Basic Auth

- Standard authentication mechanism supported by HTTP servers
- Easy to implement

```
Basic base64Encode(username:password)
```

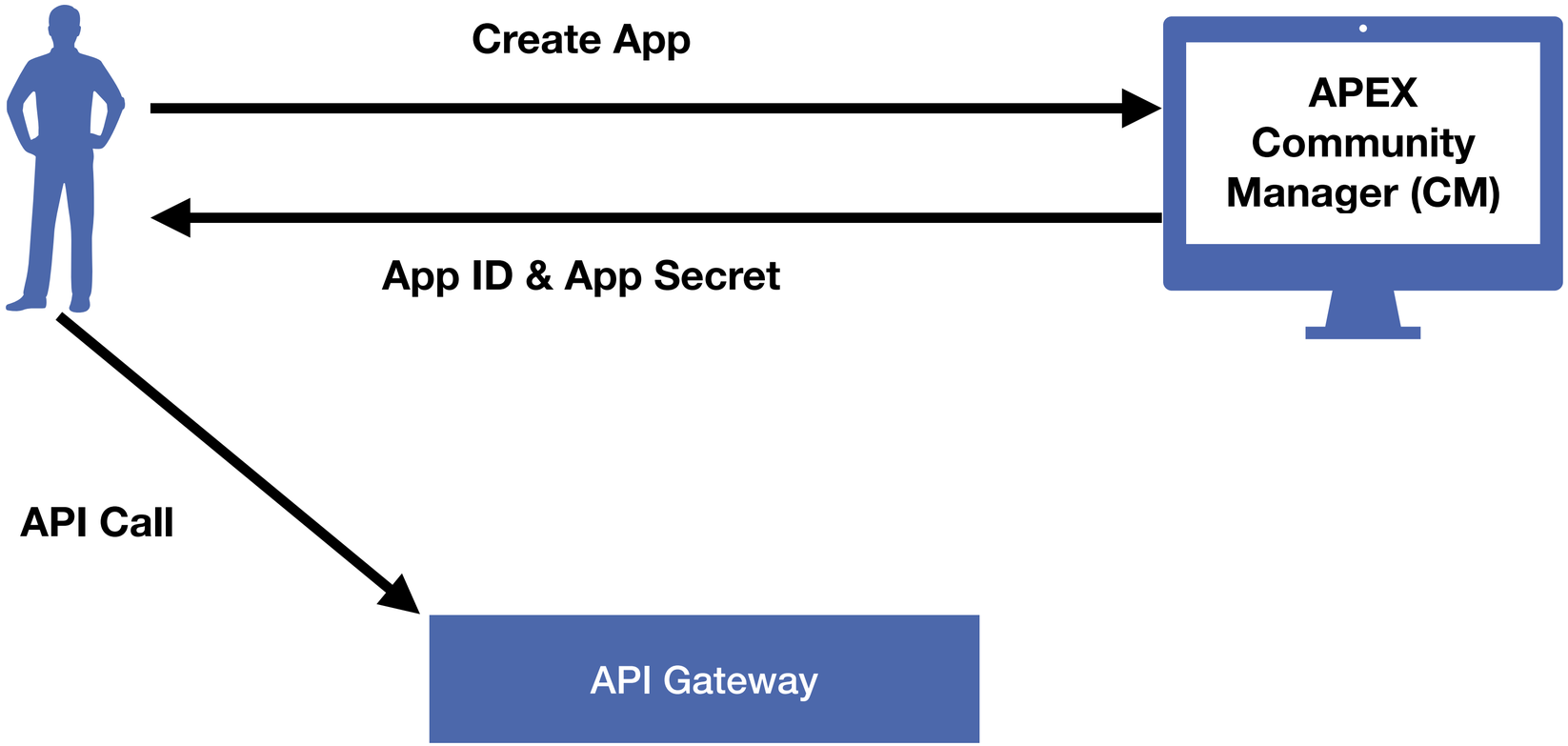
- Credential are transferred in base64-encoded plain text **(Use it with HTTPS!)**

```
GET /api/resource/12345 HTTP/1.1
Host: localhost:8080
Authorization: Basic pm9obkBzbWl0aC5jb206MTIzNDU2Nw==
```

HTTP Signature

- Protect contents send over HTTPS with a digital signature
- Provide an additional layer of protection in case HTTPS is breached
- Ensure data integrity
- Validate the API request’s authenticity
- Identify the signature’s signer (API consumer)

APEX Credentials



HTTP Signature : APEX L1

- Signature obtained through the **HMAC-SHA256** algorithm

What you would need to setup in APEX?

- App ID
- App Secret
- Generate the signature and set it to the Authorization header of a HTTP

```
GET /api/resource/12345 HTTP/1.1
Host: localhost:8080
Authorization: Apex_L1_IG realm="http://
host",apex_l1_ig_timestamp="1505727211615",apex_l1_ig_nonce="150360084
8763",apex_l1_ig_app_id="someid",apex_l1_ig_signature_method="HMACSHA2
56",apex_l1_ig_version="1.0",apex_l1_ig_signature="40Tiq5kTdt/
7QzY5nlRq3Vtd7A9qDPFJG3KUCjq5mtY="
```

HTTP Signature : APEX L2

- Signature obtained through the **RSA-SHA256** algorithm

What you would need to setup in APEX?

- App ID
- X.509 Certificate (Public Key) registered in APEX

- Generate the signature and set it to the Authorization header of a HTTP

```
GET /api/resource/12345 HTTP/1.1
Host: localhost:8080
Authorization: Apex_L2_IG realm="http://
localhost",apex_l2_ig_timestamp="1505727686138",apex_l2_ig_nonce="150662180
5283",apex_l2_ig_app_id=”someid",apex_l2_ig_signature_method="SHA256withRSA
",apex_l2_ig_version="1.0",apex_l2_ig_signature="vD1c5PpttiageSUY0n6YXTnvz0
ZlnNXQDChs+Vkxib/mbIUSQg5aEJ+mH432f4t/qL/
pC22P7G2mj4qCAfLMq1LzBP2hv+B0vAhtqNx8FjpPpbP8LrG6dB6G..."
```

govtech stack

JOSE - Javascript Object Signing and Encryption

- Defined the standard protecting the data's integrity in a JWT (JSON Web Token)
- The goal is to sign and/or encrypt the **outgoing** response message contents from APEX
- Ensure that response are not tampered with or shown to unauthorised parties

JOSE Types

- **JSON Web Encryption**
  - Encrypt the response content using Public Key Infrastructure
  - Ensure that the response **cannot be view (confidentiality)** by third party
  - Required to be attached with APEX L2 (RSA256) policy
- **JSON Web Signature**
  - Sign the response content using Public Key Infrastructure
  - Ensure that the response **cannot be modified (integrity)** by third party
  - Required to associate X.509 Certificate to the service endpoint.

Utilities

- **APEX Auth Signature Generation**
  - **Node.js Library** (https://github.com/GovTechSG/node-apex-api-security)
  - **Java Library** (https://github.com/GovTechSG/java-apex-api-security)
  - **C# Library** (https://github.com/GovTechSG/csharp-apex-api-security)
- **Interactive Signature Generator**
  - https://github.com/GovTechSG/apex-signature-validator

Demo

https://govtechsg.github.io/apex-signature-validator/

Further Reading

Sanitising information in HTTP Request

- **HTTP POST/PUT/PATCH**
  - Sensitive request data should be transferred in the request body or header
- **HTTP GET**
  - Sensitive request data should be transfer in the request header

OK:
https://example.com/resources/<id>/action

NOT OK:
https://example.com/controller/<id>/action?apiKey=a53f435643de32 (API Key in URL)

Error message with too much details

Respond with generic error message , and not too much details

OK:

```
{
  "status":  "Error",
   "code":  "401",
  "message":  "Unauthorized"
}
```

NOT OK (Too Much Details)

```
{
  "status":  "Error",
   "code":  "401",
  "message":  "Private Key : com.sun.net.ssl.internal.ssl.JSA_RSAPrivateKey@415de6
Public Key :com.sun.net.ssl.internal.ssl.JSA_RSAPublicKey@7bd9f2
Exception in thread "main" java.lang.NoClassDefFoundError: xjava/security/Cipher/
RSA256"
}
```

Other Good Practices

- Secure and whitelist HTTP methods
- Use appropriate HTTP status code for response
- Input validation at front-end as a first line of defense
- Avoid exposing management or admin endpoints
- Validate content-type by comparing request content and content type specify in the HTTP Header

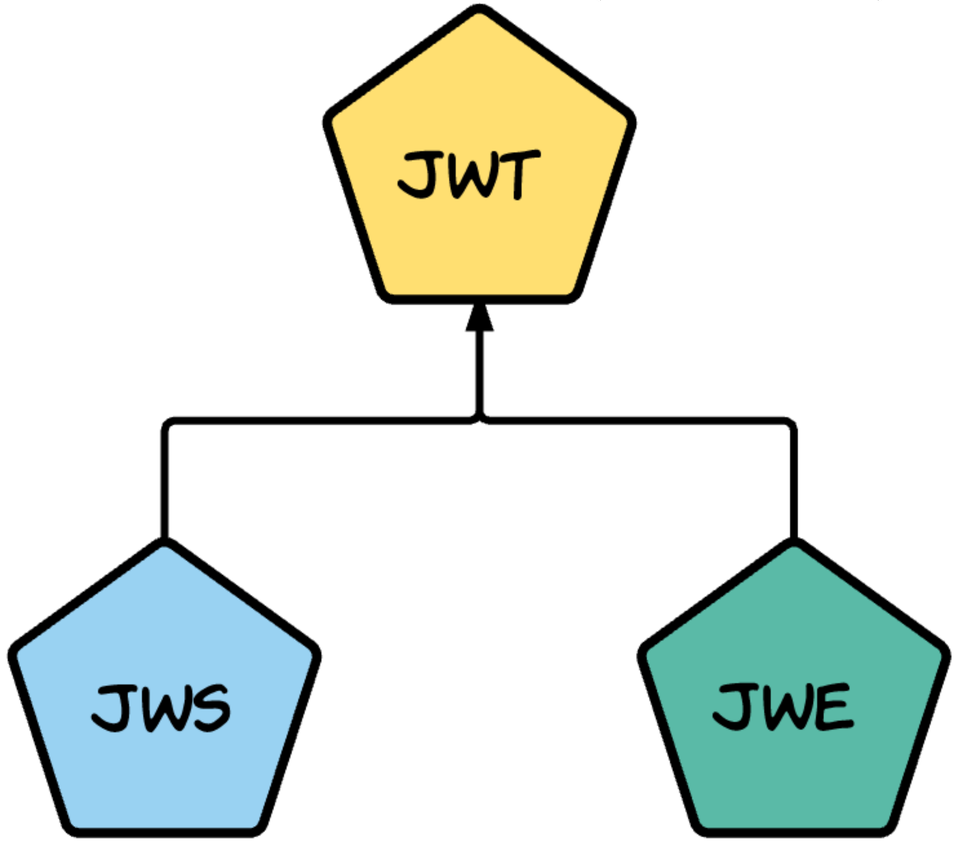
Read more on
1. https://www.owasp.org/index.php/REST_Security_Cheat_Sheet a
2. https://dzone.com/articles/top-5-rest-api-security-guidelines

JSON Web Token (JWT)

- JWT define a container to transport data and and representing claims between interested party
- The claims in a JWT are encoded as a JSON object that is digitally signed using JSON Web Signature (JWS) and/or encrypted using JSON Web Encryption (JWE).
- **Purpose**
  - Propagate and assert identity between parties
  - Transfer data securely between parties

Read more on https://medium.facilelogin.com/jwt-jws-and-jwe-for-not-so-dummies-b63310d201a3

JSON Web Token (JWT)



Relationship of JWT and JOSE

Read more on https://medium.facilelogin.com/jwt-jws-and-jwe-for-not-so-dummies-b63310d201a3