



I ❤️ APIS

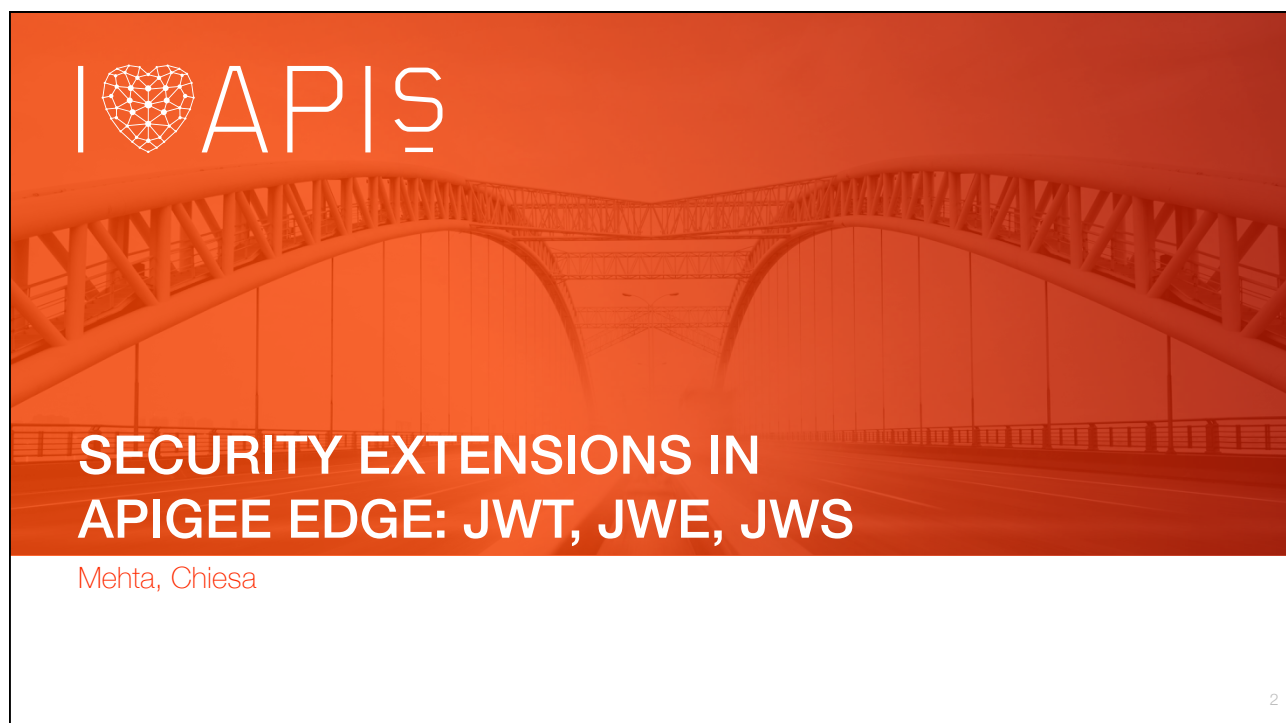
**ADVANCED SECURITY EXTENSIONS IN  
APIGEE EDGE:  
JWT, JWE, JWS**

---

Dino Chiesa,  
Vinit Mehta

apigee

1



I ❤️ APIS

**SECURITY EXTENSIONS IN  
APIGEE EDGE: JWT, JWE, JWS**

Mehta, Chiesa

2



A collection of logos for various companies and standards. At the top left is the Azure Active Directory logo, featuring a blue diamond with a white network icon. To its right is the Google logo in its multi-colored font. Below Azure Active Directory is the Salesforce logo, which is a blue cloud with the word 'salesforce' in white. In the center is the OpenID Connect logo, consisting of a grey 'C' with a yellow bar and the text 'OpenID Connect'. To the right of OpenID Connect is the Ping Identity logo, which is a red square with the word 'Ping' in white and 'Identity.' in smaller text below it. Below Salesforce is the GSMA logo, which is a black square with three red bars of increasing height and the text 'GSMA' in white. To the right of GSMA is the PayPal logo, which is a blue 'P' followed by the word 'PayPal' in blue. The entire collection of logos is set against a white background within a rectangular frame.

What do these companies have in common?

apigee

All are supporting OpenID Connect and JWT.

apigee

Authentication and Authorization is hard.

Many systems do it poorly. (Do *YOU* provide 2FA ?)

JWT and OpenID Connect will help solve that problem.

You need to get JWT, now.

apigee

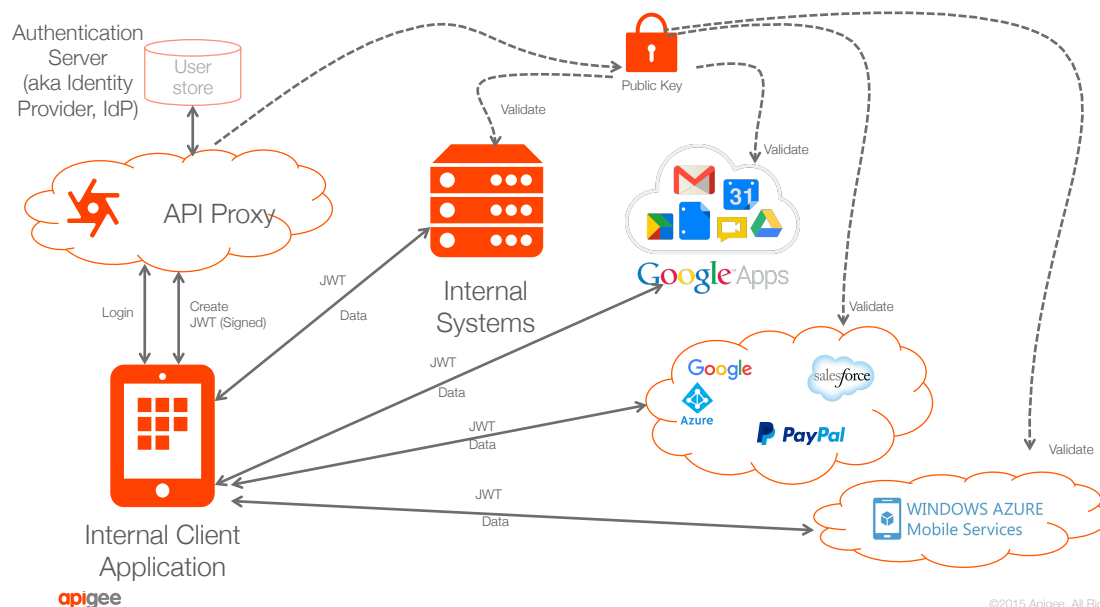


JWT, JWE, JWS

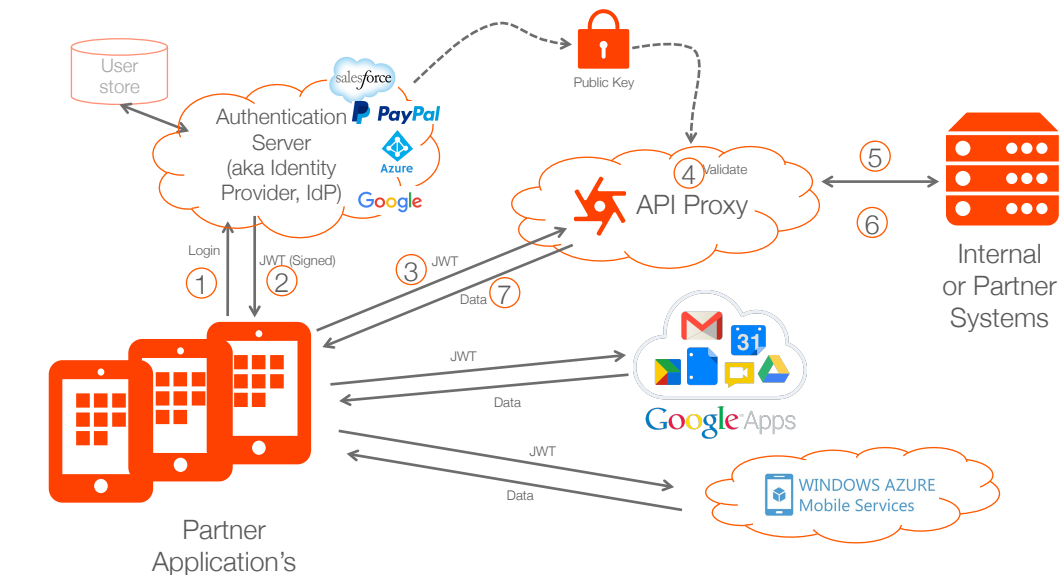
JWS, JWE, JWT are all part of JOSE:  
“JSON Object Signing and Encryption”

apigee

## JWT Enables Federated Identity



## JWT Integrates with external Identity Providers



apigee

©2015 Apigee. All Rights Reserved.

9

## JSON Web Token, Signature, Encryption

- **JWT – Token**  
IETF RFC 7519  
<https://tools.ietf.org/html/rfc7519>
- **JWS – Signature**  
IETF RFC 7515  
<https://tools.ietf.org/html/rfc7515>
- **JWE – Encryption**  
IETF RFC 7516  
<https://tools.ietf.org/html/rfc7516>
- Signed, or optionally, Encrypted, set of claims.
- Issuer, Subject, audience, issue time, not-before time, expiration.
- Used as a BEARER token
- “Self-validating”
- Receiving parties can make decisions based on the claims and signing party.
- JSON representation of Signed or HMAC’ed Content
- Payload that is signed *need not be JSON!*
- The resulting JWS can be verified by receivers
- JSON representation of Encrypted content
- Payload that is encrypted *need not be JSON*
- Resulting JWE Can be decrypted by receivers

apigee

©2015 Apigee. All Rights Reserved.

10

## JSON Web Token, Signature, Encryption

- JWT – Token  
IETF RFC 7519  
<https://tools.ietf.org/html/rfc7519>
  - Signed, or optionally, Encrypted, set of claims.
  - Issuer, Subject, audience, issue time, not-before time, expiration.
  - Used as a BEARER token
  - “Self-validating”
  - Receiving parties can make decisions based on the claims and signing party, or encrypting party.
- JWS – Signature  
IETF RFC 7515  
<https://tools.ietf.org/html/rfc7515>
  - JSON representation of Signed or HMAC’ed Content
  - Payload that is signed *need not be JSON!*
  - The resulting JWS can be verified by receivers
- JWE – Encryption  
IETF RFC 7516  
<https://tools.ietf.org/html/rfc7516>
  - JSON representation of Encrypted content
  - Payload that is encrypted need not be JSON
  - Resulting JWE Can be decrypted by receivers



Demo: Azure AD JWT  
and JWT.io

Apigee Edge includes standard policies for many security tasks.

OAuth1.0a generation and verification,  
OAuth2 generation and verification,  
SAML generation and verification...

apigee

Apigee Edge does not yet include standard policies for  
JWT, JWE, JWS

apigee

But ... Code + Configure !

apigee

## What are Java Callouts?

- Embed your Java code as a policy in Apigee Edge
- One Interface, one method, 2 parameters
- Can read policy configuration
- Can read and write context variables
- ...anchor anywhere in Edge policy flow
- One of the ways to extend Edge with custom code. Also JavaScript, Python, nodejs.
- RTFM:  
<http://apigee.com/docs/api-services/reference/java-callout-policy>

```

/Users/dino/dev/java/callouts/hmac/src/com/dinochiesa/hash/HMAC_Creator_Callout.java [mod
public ExecutionResult execute(MessageContext msgCtx,
                               ExecutionContext exeCtx) {
    try {
        String SIGNING_KEY = getKey(msgCtx);
        String MESSAGE = getMessage(msgCtx);
        String algorithm = getAlgorithm(msgCtx);
        msgCtx.setVariable("hmac.alg", algorithm);

        String javaizedAlg = javaizeAlgorithmName(msgCtx.algorithm);
        msgCtx.setVariable("hmac.javaizedAlg", javaizedAlg);

        Mac hmac = Mac.getInstance(javaizedAlg);
        SecretKeySpec key = new SecretKeySpec(SIGNING_KEY.getBytes(), javaizedAlg);
        hmac.init(key);

        String signature = Hex.encodeHexString(hmac.doFinal(MESSAGE.getBytes("UTF-8")));

        msgCtx.setVariable("hmac.key", SIGNING_KEY);
        msgCtx.setVariable("hmac.stringToSign", MESSAGE);
        msgCtx.setVariable("hmac.alg", algorithm);
        msgCtx.setVariable("hmac.signature", signature);

    } catch (Exception e) {
        e.printStackTrace();
        msgCtx.setVariable("hmac.error", "Exception " + e.toString());
        return ExecutionResult.ABORT;
    }
    return ExecutionResult.SUCCESS;
}

```

apigee

©2015 Apigee. All Rights Reserved.

16



## Java Callout for JWT Parse/Verification

- Re-usable now in any of your Proxies
- Configure it with XML as any other policy
- Make decisions based on embedded claims
- Can read JWT generated by third parties, such as Google or Windows Azure

```

/Users/dino/dev/Advanced-Security-Extensions-1/repo/jwt_signed/apiproxy/apiproxy/policie...
<JavaCallout name='JavaCallout-JWT-Parse-HS256'>
  <Properties>
    <Property name='algorithm'>HS256</Property>
    <Property name='jwt'>{request.formparam.jwt}</Property>
    <Property name='secret-key'>{request.formparam.key}</Property>

    <!-- verify these claims -->
    <Property name='claim_sub'>jwt_signed</Property>
    <Property name='claim_iss'>http://dinochiesa.net</Property>
    <Property name='claim_aud'>Optional-String-or-URI</Property>
    <Property name='claim_shoesize'>9</Property>

  </Properties>

  <ClassName>com.apigee.callout.jwtsigned.JwtParserCallout</ClassName>
  <ResourceURL>java://jwt-signed-edge-callout.jar</ResourceURL>
</JavaCallout>

```

<https://github.com/apigee/iloveapis2015-jwt-jwe-jws>

## Java Callout for JWT Generation

- Re-usable now in any of your Proxies
- Configure it with XML as any other policy
- Generate JWT for use by others
- Can be used by backends or other systems called by clients
- Can be consumed by Edge itself

```

/Users/dino/dev/Advanced-Security-Extensions-1/repo/jwt_signed/apiproxy/apiproxy/policie...
<JavaCallout name='JavaCallout-JWT-Create-RS256-2' >
  <DisplayName>JavaCallout-JWT-Create-RS256-2</DisplayName>
  <Properties>
    <Property name='algorithm'>RS256</Property>

    <!-- private-key and private-key-password used only for algorithm = RS256 -->
    <Property name='private-key'>
      -----BEGIN RSA PRIVATE KEY-----
      sMuDK2Zg+TRJ+nZbdIjhg1VLzMLPv3MCxhlji7H4y8YIVD738rpJLOY3LBqh9ilo
      ....
      7Z0F1UXVaoldDs+izZo5biVF/NNIBtg2FkZd4hh/cFLf1PV+M5+SmA==
      -----END RSA PRIVATE KEY-----
    </Property>

    <!-- standard claims -->
    <Property name='subject'>{apiproxy.name}</Property>
    <Property name='issuer'>http://dinochiesa.net</Property>
    <Property name='audience'>Optional-String-or-URI</Property>
    <Property name='expiresIn'>86400</Property> <!-- in seconds -->

    <!-- custom claims -->
    <Property name='claim_primarylanguage'>English</Property>
    <Property name='claim_shoesize'>8.5</Property>
    <Property name='claim_motto'>Iloveapis</Property>

  </Properties>

  <ClassName>com.apigee.callout.jwtsigned.JwtCreatorCallout</ClassName>
  <ResourceURL>java://jwt-signed-edge-callout.jar</ResourceURL>
</JavaCallout>

```



### Some comments

- This JWT policy handles Signed, not Encrypted JWT
- RS256 and HS256 are supported
- We have a different policy that produces Encrypted JWT (JWE) using RS256
- JWT cannot be “revoked” – so limit your lifetimes
- Exercise for the reader:
  - ES256, other algorithms

## When to use JWT vs Oauth 2.0 tokens?

apigee

## When to use JWT vs Oauth 2.0 tokens?

- Trick Question! *JWT are OAuth2.0 tokens*
- Better phrased as: **When to use JWT vs Opaque Oauth 2.0 tokens?**
- Federation
- When you want the client to know everything that is being claimed
- JWT implies minimal impact to client and server apps
- JWT do not work well with revocation

apigee

©2015 Apigee. All Rights Reserved.

22

## Java Callout for JWE Generation

- Re-usable now in any of your Proxies
- Configure it with XML as any other policy
- Generate JWE for use by others
- Configurable Key strength and key derivation
- Can be used by backends or other systems called by clients
- Can be consumed by Edge itself

```

<JavaCallout name='JavaCallout-JWE-Encrypt-A128CBC-HS256' >
  <Properties>
    <Property name='algorithm'>A128CBC-HS256</Property>
    <Property name='secret-key'>{request.formparam.key}</Property>
    <Property name='plaintext'>{request.formparam.plaintext}</Property>
  </Properties>

  <ClassName>com.apigee.callout.jwe.JweEncryptorCallout</ClassName>
  <ResourceURL>java://jwe-edge-callout.jar</ResourceURL>
</JavaCallout>

```

<https://github.com/apigee/iloveapis2015-jwt-jwe-jws>

## Java Callout for JWE Decryption

```

<JavaCallout name='JavaCallout-JWE-Decrypt-A128CBC-HS256' >
  <Properties>
    <Property name='algorithm'>A128CBC-HS256</Property>
    <Property name='secret-key'>{request.formparam.key}</Property>
    <Property name='jwe'>{request.formparam.jwe}</Property>
  </Properties>

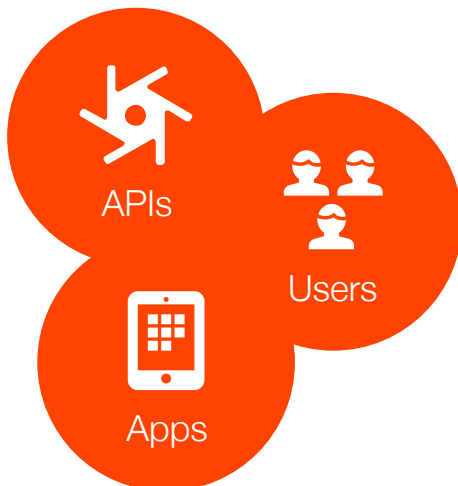
  <ClassName>com.apigee.callout.jwe.JweDecryptorCallout</ClassName>
  <ResourceURL>java://jwe-edge-callout.jar</ResourceURL>
</JavaCallout>

```

## When to use JWS and JWE?

- Trick Question! *Don't ever use them!*
- *No, seriously.*
- JWS and JWE imply some change to client apps
- More limited scope of usage than JWT
- There are already ways to sign and encrypt arbitrary data
- My opinion: JWE and JWS are mostly interesting in support of JWT

## What did we learn?



- YOU NEED to handle JWT
- You can use JWT, JWS, JWE in Apigee Edge today via custom policies
- No coding needed !
- These policies complement the existing built-in policies in Apigee Edge

<https://github.com/apigee/iloveapis2015-jwt-jwe-jws>