JWTs and Their Application

Martin Snyder, CTO Wingspan Technology

Philly JUG - 15-Feb-2017

Agenda

- What are JWTs
- Anatomy of a JWT
- Mechanics of using JWTs
- Applications of JWTs

What are JWTs

- JSON Web Token
- Mechanism for transferring verifiable claims
- Base64 encoded JSON

Anatomy of a JWT 1/3

HEADER.CLAIMS.SIGNATURE

Anatomy of a JWT 2/3

eyJhbGciOiJIUzUxMiJ9.
eyJzdWIiOiJtc255ZGVyK2FiY2RAd2luZ3NwYW4u
Y29tIiwiZXhwIjoxNDg5NzYxNTIwfQ.

5FMZQACCtVWkIEDcbAXc4R3LQmjTq6O2kYtiBz
9SWeOW8U0MG-3Up7KRrnByVlu-
u5UnDFtpCTQO9S7wVQLEBw

Anatomy of a JWT 3/3

{"alg":"HS512"}.

{“sub":"msnyder@wingspan.com", "exp":
1489761520}.

e4 53 19 40 00 82 b5 55 a4 94 40 dc 6c 05 dc e1
96 f1 4d 0c 1b 75 29 ec a4 6b 9c 1c 95 96 eb b9
1d cb 42 68 d3 ab a3 b6 91 8b 62 07 3f 52 59 e3
52 70 c5 b6 90 93 40 ef 52 ef 05 50 2c 40 70

Headers

- alg - Signature algorithm
- typ - Token Type
- zip - Compression algorithm
- Others - Part of JWE and JWS standards

Claims

- iss - Issuer
- sub - Subject
- aud - Audience
- exp - Expiration
- nbf - Not Before
- iat - Issued At
- jti - JWT ID (unique identifier)

Signature

- Signs HEADER.CLAIMS

Mechanics of JWT

- Use JJWT (https://github.com/jwtk/jjwt) for Java
- Developed and maintained by Stormpath (https://stormpath.com/)

Algorithms

HS256: HMAC using SHA-256

HS384: HMAC using SHA-384

HS512: HMAC using SHA-512

ES256: ECDSA using P-256 and SHA-256

ES384: ECDSA using P-384 and SHA-384

ES512: ECDSA using P-521 and SHA-512

Creating a JWT

```
import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.impl.crypto.MacProvider;

import java.security.Key;
```

```
// We need a signing key, so we'll create one just for this example. Usually

// the key would be read from your application configuration instead.

Key key = MacProvider.generateKey();


String compactJws = Jwts.builder()

  .setSubject("Joe")

  .signWith(SignatureAlgorithm.HS512, key)

  .compact();
```

Verifying a JWT

```
try {
    Jws<Claims> claims = Jwts.parser()
        .requireSubject("Joe")
        .require("hasMotorcycle", true)
        .setSigningKey(key)
        .parseClaimsJws(compactJws);
} catch (MissingClaimException e) {
    // we get here if the required claim is not present
} catch (IncorrectClaimException) {
    // we get here if the required claim has the wrong value
}
```

Definite DO

- Use existing key names where possible
- Use NBF and EXP, or at least EXP
- Validate the **existence** of expected claims
- Respect the secret key

Definitely DO NOT

- Examine JWT contents without validating
- Overload the JWT
- Compromise key security

Application #1 - User Verification

1. User fills out a registration form
2. Server encodes registration information in a JWT
3. Server emails a link to the provided email address containing the JWT
4. User clicks the link
5. Server creates the user record in the database

Application #2 - Trusted Authentication

- Serialize authenticated runtime objects as JWT
- Pass JWT from one application to another
- Use Issuer (iss) and Audience (aud) claim to refine the JWT scope
- Use a Public/Private key pair algorithm
- Works across tiers, platforms and languages

JWT Usage Goals

- Limit Database interaction
- Limit transient Database entries

JWT Challenges

- JWT Revocation
- Algorithm, hash-width, and secret selection
- Resisting the urge to JWT everything

Links

Wikipedia: https://en.wikipedia.org/wiki/JSON_Web_Token

Specifications:

https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-32

https://tools.ietf.org/html/draft-ietf-jose-json-web-signature-41

https://tools.ietf.org/html/draft-ietf-jose-json-web-encryption-40

More links

JJWT - https://github.com/jwtk/jjwt

Micah Video - https://www.youtube.com/watch?v=c-6hv70h9mc

My Blog - http://martinsnyder.net/

Twitter - https://twitter.com/MartinSnyder

Thank You!

Martin Snyder, CTO Wingspan Technology
Philly JUG - 15-Feb-2017