

PREGRADO



UNIDAD 3 | WEB SERVICES

WEB SERVICES SECURITY

PART 1

Al finalizar la semana, el estudiante ejecuta un proceso ágil para el desarrollo una aplicación de lado servidor, documentada, verificada y validada, con características innovadoras, bajo una arquitectura orientada a servicios y aplicando los principios RESTful utilizando el lenguaje C# y Microsoft .NET Framework.

AGENDA

JWT

CORS

ASP.NET CORE & SECURITY



JSON Web Token

JSON Web Token (JWT) es un open standard (RFC 7519) que define una manera compacta y autocontenida para transmitir de forma segura información entre parties a manera de un JSON object.

Esta información es confiable y verificable dado que está firmada digitalmente.

JWTs pueden firmarse usando un secret (con el algoritmo HMAC) o un public/private key pair usando RSA ó ECDSA.

Cuándo usar JSON Web Tokens

Escenarios de uso:

Authorization. El escenario más común. Una vez que el usuario inicia sesión, cada request posterior incluirá el JWT, permitiendo al usuario acceder a routes, services y resources que se estén permitidos para ese token.

Information Exchange. JSON Web Tokens son una manera de transmitir información segura entre parties. Dado que los JWTs se pueden firmar se puede estar seguro de que los senders son quienes dicen ser. Adicionalmente, dado que la firma se calcula usando el header y payload, puede verificarse que el contenido no ha sido manipulado.

Structure

En su forma compacta, JSON Web Tokens consisten de tres partes separadas con puntos (.):

Header

Payload

Signature

xxxxxx.yyyyyy.zzzzzz

Header

El header normalmente consta de dos partes: tipo de token, que es JWT y el signing algorithm que se usa, como HMAC SHA256 o RSA. Este JSON tiene un encoding usando Base64Url para la primera parte del JWT.

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

Payload

Contiene los claims. Claims son los statements acerca de un entity (normalmente user) y datos adicionales.

Hay tres tipos de claims: **registered**, **public** y **private claims**.

Registered claims. Claims predefinidos no obligatorios pero recomendados, para proporcionar un conjunto de claims que sea útil e interoperable. Entre ellos están iss (issuer), exp (expiration time), sub (subject), aud (audience), entre otros.

Public claims. Definidos a voluntad por los que usen JWTs. Para evitar colisiones deberían estar definidos en un IANA JSON Web Token Registry o estar definidos como URI que contenga un collision resistant namespace.

Private claims. Claims personalizados creados para compartir información entre parties que acuerdan usarlos y no son ni registered ni public claims.

Payload

Por ejemplo

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

Este payload luego es encoded con Base64Url para formar la segunda parte del JSON Web Token.

Signature

Para crear el signature se toma el header, encoded payload, un secret, el algorithm especificado en el header y se firma.

Por ejemplo, usando el HMAC SHA256 algorithm se crea un signature de esta manera.

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

El signature permite verificar que el mensaje no ha sido alterado en el camino. En caso de signed tokens con private key, permite verificar que el sender del JWT es quien dice ser.

All together

Un JWT con un header y payload que están encoded y signed con

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4  
gRG91IiwiaXNTb2NpYWwiOi0nRydWV9.  
4pcPyMD09o1PSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Para comprobar los conceptos

<https://jwt.io/#debugger-io>

JSON Web Tokens in action

Cuando el usuario inicia sesión con éxito usando sus credenciales, se retorna un JSON Web Token.

Cuando el usuario desea acceder a routes o resources protegidos, el user agent debería enviar el JWT, normalmente en el **Authorization** header usando el esquema **Bearer**.

```
Authorization: Bearer <token>
```

AGENDA

JWT

CORS

ASP.NET CORE & SECURITY



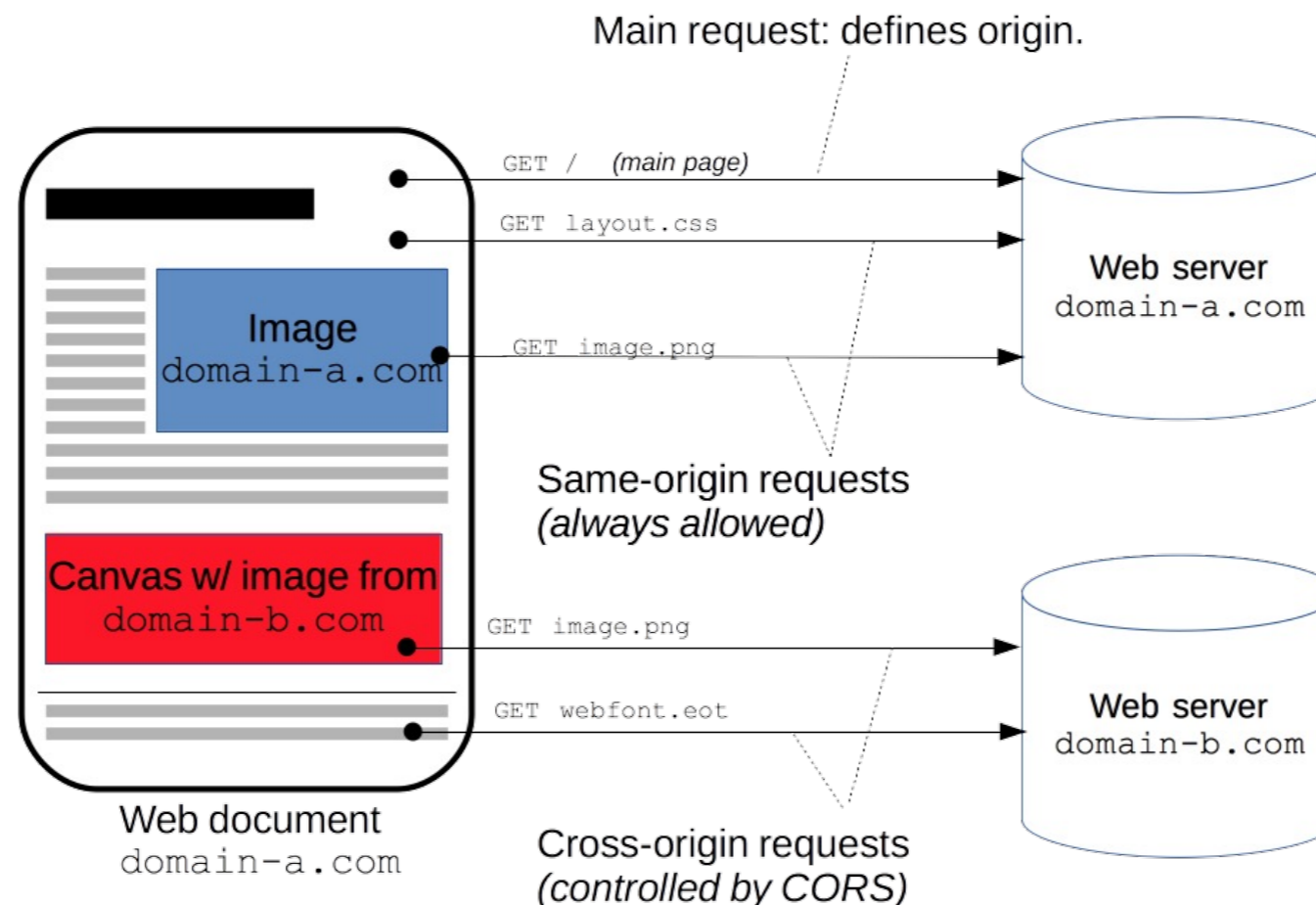
Cross-Origin Resource Sharing (CORS)

Mecanismo basado en HTTP-header que permite a un servidor indicar a cualquier otro origin (domain, protocol, port) distinto del suyo, desde el cual un browser debería permitir la carga de resources.

Los browsers hacen un preflight request hacia el server que tiene alojado el cross-origin resource, para ver si el server permitirá el request real. En dicho preflight, el browser envía headers que indican el HTTP method y headers que se usarán en el request real.

Cross-Origin Resource Sharing (CORS)

Ejemplo: frontend JavaScript code en <https://domain-a.com> usa XMLHttpRequest para hacer un request a <https://domain-b.com/data.json>.



AGENDA

JWT

CORS

ASP.NET CORE & SECURITY



Microsoft Identity Model Tokens

`Microsoft.IdentityModel.Tokens`

Incluye tipos que brindan soporte para SecurityTokens, Cryptographic operations como Signing, Verifying Signatures, Encryption.

`System.IdentityModel.Tokens.Jwt`

Incluye tipos que brindan soporte para crear, serializar y validar JSON Web Tokens.

Microsoft ASP.NET Core Authentication

`Microsoft.AspNetCore.Authentication.JwtBearer`

ASP.NET Core middleware que permite que una aplicación reciba un OpenID Connect bearer token.

ASP.NET Core and CORS

Habilitar CORS en ConfigureServices.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors();
}
```

Establecer opciones de CORS en Configure.

```
// CORS Configuration
app.UseCors(x => x
    .SetIsOriginAllowed(origin => true)
    .AllowAnyMethod()
    .AllowAnyHeader()
    .AllowCredentials());
```

RESUMEN

Recordemos

JWT es un método abierto, basado en el industry standard RFC 7519 para representar claims de forma segura entre dos partes.

CORS es una especificación que permite acceso abierto entre límites de dominio.



REFERENCIAS

Para profundizar

<https://jwt.io/>

<https://docs.microsoft.com/en-us/dotnet/api/system.identitymodel.tokens.jwt?view=azure-dotnet>

https://www.w3.org/wiki/CORS_Enabled

<https://docs.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-5.0>



PREGRADO

Ingeniería de Software

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería



UPC

Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco
Lima 33 - Perú
T 511 313 3333
<https://www.upc.edu.pe>

exígete, innova