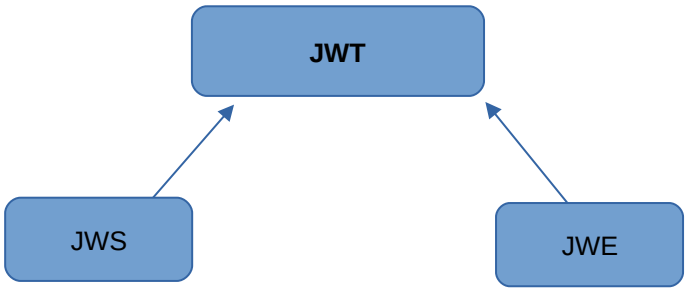
HINTS

ESTÁNDARES PARA LA DEFINICIÓN DE TOKENS

En este apartado se analizarán los diferentes estándares que existen para la definición de seguridad para los tokens que utilizaremos en las API REST.

- JSON Web Tokens (JWT) (Wikipedia 2018): estándar basado en el formato del tipo JSON que permite la creación de tokens permitiendo transmitir información como por ejemplo identidad, roles, privilegios.
- JSON Web Signature (JWS) (Wikipedia 2017): estándar propuesto para la firma de un JWT que nos permitirá validar la información junto con una "palabra secreta" indicando que el mensaje enviado no ha sido alterado desde que se firmó.
- JSON Web Encryption (JWE) (Jones 2015): nos permite encriptar el contenido que se envía previniendo que terceros tengan acceso a la información que se está transmitiendo.
- Open Authorization 2.0 (OAuth) (Wikipedia 2009): es un estándar abierto para la autorización que nos permite obtener acceso limitado a sitios webs, aplicaciones móviles o aplicaciones de escritorio. Permitirá a un usuario otorgar acceso a una aplicación de terceros a los recursos protegidos del usuario, sin revelar sus credenciales o incluso su identidad.

Por realizar un símil a POO podemos indicar que JWT es una clase abstracta en donde JWS y JWE heredan las propiedades de JWT como se aprecia en la figura siguiente. Tenemos que entender que JWT por sí mismo es un concepto que explica cómo debemos de actuar a la hora de transferir los datos.



JSON OBJECT SIGNING AND ENCRYPTION (JOSE)

JWT define el formato del token y utiliza especificaciones complementarias para manejar la firma y el cifrado de una forma más básica como se podía ver en los puntos anteriores. Ahora una forma de aunar todos estos estándares es mediante una colección de especificaciones que se conoce como JOSE (JavaScript Object Signing and Encryption) y tras consultar su RFC 75208 podemos indicar que consta de los siguientes componentes:

- JSON Web Signature (JWS – RFC 7515): define el proceso para firmar digitalmente un JWT.
- JSON Web Encryption (JWE – RFC 7516): define el proceso para cifrar un JWT.
- JSON Web Algorithm (JWA – RFC 7518): define una lista de algoritmos para firmar digitalmente o cifrar.
- JSON Web Key (JWK – RFC 7517): define cómo se representa una clave criptográfica y conjuntos de claves.

HERRAMIENTAS GATEWAY PARA API REST

Existen en el mercado una amplia gama de opciones para la gestión de las API. Herramientas que nos permiten funcionalidades como la monitorización del tráfico, aplicar políticas de seguridad, políticas de consumo, rendimiento, cuadro de mandos (dashboard), etc. Entre estas se encuentran las siguientes:

- **KONG API GATEWAY**: proyecto open-source que proporciona una capa de abstracción flexible que gestiona de forma segura la comunicación entre clientes y microservicios a través de su API. Kong se suele instalar con Konga 2 para que el manejo de las configuraciones de las API sea más cómodo y ágil para el usuario administrador.
- **TYK API GATEWAY**: proyecto open-source que proporciona rapidez y escalabilidad, su plataforma permite la gestión de API de forma gráfica a través su panel web.
- **APIGEE API GATEWAY**: proyecto adquirido por Google en 2016. No es de código abierto y se basa en Java empresarial. Inicialmente comenzaron como una aplicación XML / SOA, pero pasaron al espacio de administración de API.

OTRAS HERRAMIENTAS

- API REST Pública
- La API REST del sitio web https://reqres.in/ nos permitirá realizar pruebas con una API real. Se ajusta a los principios REST y simula escenarios de aplicaciones reales, como probar un sistema de autenticación de usuarios.
- Swagger Editor
- Se utilizará esta herramienta para generar la documentación de la API. El editor proporciona ayuda visual para las especificaciones de OpenAPI, maneja errores y proporciona autocompletados en tiempo real. https://editor.swagger.io/
- RoboMongo, Rabo 3T
- Con esta herramienta podremos administrar de forma gráfica las bases de datos de mongoDB. https://robomongo.org/
- jwt-decoder
- Extensión para Visual Studio Code (VSC) que permite decodificar cadenas JWT. https://marketplace.visualstudio.com/items?itemName=jflbr.jwt-decoder

DECODIFICANDO Y VALIDANDO TOKENS

Para decodificar o averiguar la información contenida en un token, puede usar cualquier servicio de validación de JWT que pueda encontrar en línea. Por ejemplo, algunos sitios populares son los siguientes:

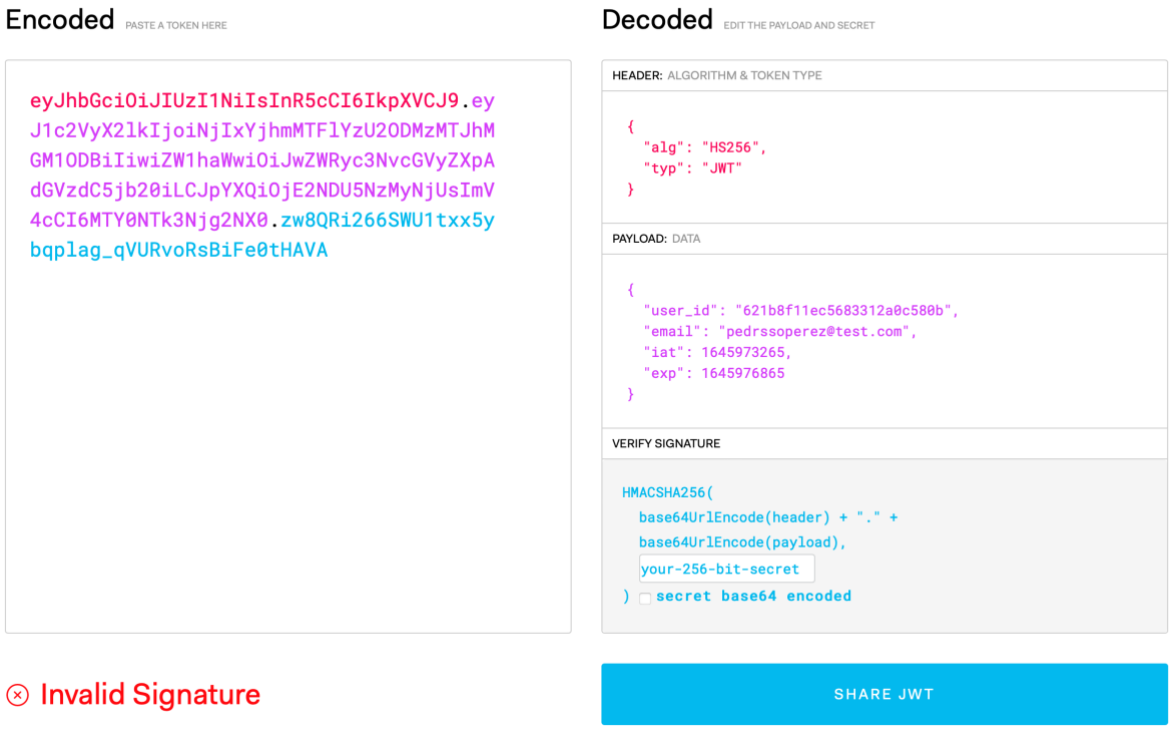
- jwt.io
- token.dev
- pingidentity

Aunque todos estos servicios tienen diferentes interfaces de usuario, en esencia, todos funcionan de la misma manera, uno ingresa el JWT y el sitio devuelve todos los detalles contenidos en el token. Para efectos de ejemplificación, vamos a usar el mismo token generado a partir de nuestro Exercise:

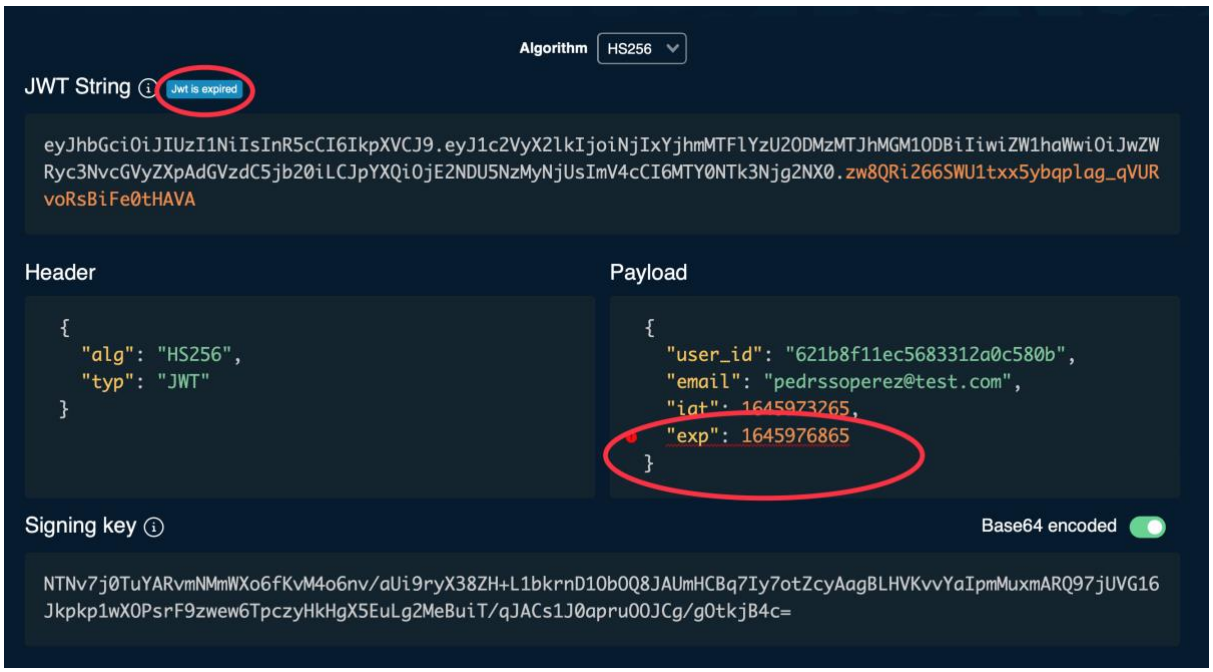
```
1 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoiNjIxYjhmMTFlYzU2ODM
2 zMTJhMGM1ODBiIiwiZW1haWwiOiJwZWRyc3NvcGVyZXpAdGVzdC5jb20iLCJpYXQiOjE2NDU
3 5NzMyNjUsImV4cCI6MTY0NTk3Njg2NX0.zw8QRi266SWU1txx5ybqplag_qVURvoRsBiFe0t
4 HAVA
```

Copiemos nuestro token y peguémoslo en cada uno de los sitios mencionados anteriormente para mostrar cómo cada sitio se diferencia entre sí.

En jwt.io nos muestra el contenido del token y en la parte inferior izquierda nos indica si el token está válido o no:



En token.dev también encontramos información similar y aparte, se nos indica si el token expiró o no:



En pingidentity también recibimos la información del contenido del token y aparte, podemos verificar la firma del token con nuestra clave secreta que, en nuestro caso, corresponde a la información del token key. Al ingresar esta información podemos ver que al final del formulario del sitio se nos indica que la firma es válida.

```
TOKEN_KEY=d6699170151a914e2b5a67c3b913401e971f28e66427286d7e3af04194d764
fc70b14666868121c515188a9aff1fc3550d7a25d361dbd779863ee6cb9a2d7abc
```

INVOCANDO Y SERVICIO PASANDO UN TOKEN COMO PARÁMETRO

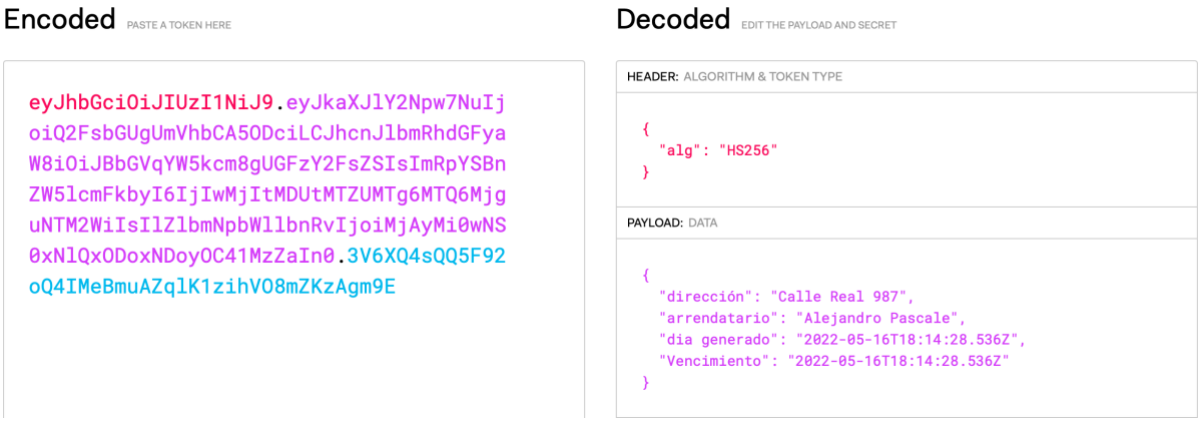
Los parámetros de URL existen desde hace mucho tiempo. Su larga historia nos enseña que son fáciles de manipular. Los atacantes pueden iniciar solicitudes con valores arbitrarios para los parámetros de URL.

El uso de identificadores aleatorios corresponde a confiar en la oscuridad por seguridad. Un mejor enfoque es garantizar la integridad de los parámetros de URL. De esa forma, cualquier manipulación por parte del atacante será detectable por la aplicación que consume los parámetros de la URL.

Hoy en día, la forma más sencilla de proporcionar una capa de seguridad a los parámetros de la URL es mediante los JSON Web Token. Los JWT proporcionan una forma de intercambiar la seguridad de las reclamaciones entre dos partes. La URL del siguiente ejemplo utiliza un JWT para transferir información de una aplicación a otra.

```
gestionador.arriendos.com?params=eyJhbGciOiJIUzI1NiJ9.eyJkaXJlY2Npw7NuIj
oiQ2FsbGUgUmVhbCA5ODciLCJhcnJlbmRhdGFyaW8iOiJBbGVqYW5kcm8gUGFzY2FsZSIsIm
RpYSBnZW5lcmFkbyI6IjIwMjItMDUtMTZUMTg6MTQ6MjguNTM2WiIsIlZlbmNpbWllbnRvIj
oiMjAyMi0wNS0xNlQxODoxNDoyOC41MzZaIn0.3V6XQ4sQQ5F92oQ4IMeBmuAZqlK1zihVO8
mZKzAgm9E
```

El JWT parece un montón de caracteres aleatorios, pero es una representación codificada en base64 de los datos. La decodificación del JWT con jwt.io muestra el contenido real del JWT.



Como puedes ver, la carga útil del JWT ahora contiene los parámetros que deseamos intercambiar. Además de los datos, el JWT también contiene un encabezado con metadatos y una firma para garantizar la integridad de los datos, que no se muestran en la versión decodificada.