JWT: Json Web Token

Ingeniería Informática
Teoría y Aplicación de la Informática 2

Joaquín Olivera

Universidad Católica "Nuestra Señora de la Asunción"

Abstract. El articulo trata de explicar esta tecnología que data del año 2014 y que es muy utilizada en el ambiente laboral, con soporte en múltiples lenguajes y frameworks. Describir algunos de los beneficios que son encontrados por los usuarios y algunos de los problemas de la misma, comparativas con tecnologías similares.

Keywords: json,token, autenticación,autorización , seguridad, servicios web, sesiones, claims

1 Introducción

Una buena forma de entender lo que este trabajo pretende señalizar es empezar por los conceptos más sencillos, que quizás algunas personas desconozcan, así que antes de ponernos en marcha con el articulo, empezamos definiendo estos conceptos, que permitirán un mayor entendimiento del tema. Dicho tema es la presentación de la herramienta JWT, muy utilizada en ambiente laboral del país, con vistas a lo que son los procesos de autenticación de usuarios antes recursos o aplicaciones que las empresas tengan en su lista de productos.Los servicios web son cada vez más comunes, y por lo tanto se van haciendo más necesario este tipo de herramientas. Hoy día también, la conectividad remota es a lo que se apunta y esto requiere los procesos de identificar a los usuarios, darle seguridad y además de clasificar los recursos que puedan ser accedidos según el tipo de usuario, típicos planes entre los planes que se ofrecen en las compañías. En todo esto, entra JWT en la actualidad.

1.1 Definiciones de términos relativos al tema

Estas son las definiciones de los conceptos que se consideran necesarias para poder tener un mejor entendimiento del trabajo que se quiere presentar a continuación.

Que entendemos al hablar de Autenticación? En palabras sencillas, el proceso de autenticación es una manera de demostrar que eres la persona/usuario que decís ser. Diferente a la autorización que es controlar el acceso de un usuario a ciertos recursos que se posean. El modelo más genérico de autenticación es cuando una persona se inscribe a algún lugar, y este le valida las credenciales, que suelen contar de un usuario y contraseña.Además de la autenticación tradicional, es bueno saber algunos otros métodos de autenticación, divididos según su clase En este caso se hace la clasificación según el modo
Basados en algo conocido (contraseña) Los sistemas de autenticación más usados son aquellos basados en conocimiento, por ejemplo los que hacen uso de un ID de usuario y una contraseña, o de una llave criptográfica.

Basados en algo poseído (tarjeta de identidad, usb, token) Sistemas de autenticación basados únicamente en algo que el usuario posee, estos métodos se basan en que el usuario tiene en su poder un token, los cuales podemos dividir en tokens de memoria y tokens inteligentes.
Basados en características físicas (voz, huellas, ojos) Para lograr la autenticación estos sistemas hacen uso de las características o atributos, fisiológicos y de comportamiento, propios de cada individuo que lo hacen único. Dichos sistemas son conocidos también como sistemas biométricos.[6]

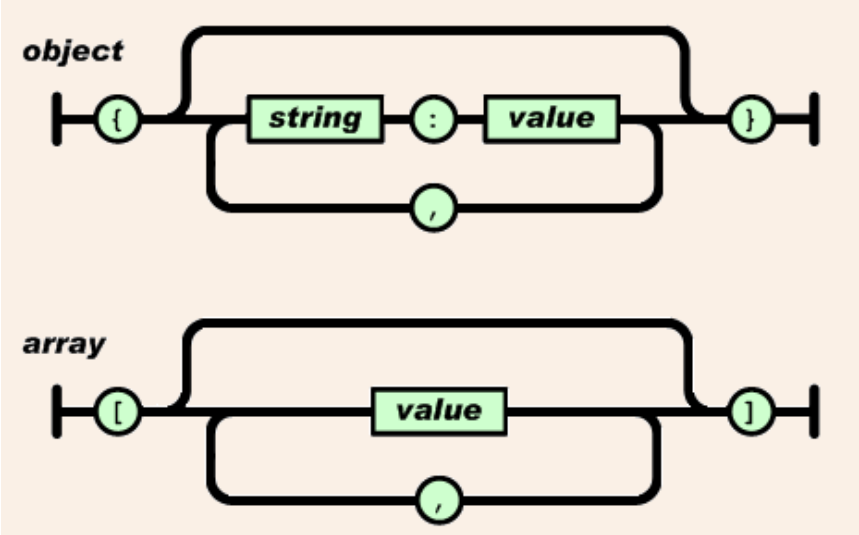
Que es un JSON? JSON es una abreviatura de JavaScript Object Notation, y es una forma de almacenar información de forma organizada y de fácil acceso. En pocas palabras, nos da una colección legible de datos que podemos acceder de una manera realmente lógica, lo caracterizan su sencillez y facilidad de uso, como así también lo compacto.[8]

```
var jason = {
        "age" : "24",
        "hometown" : "Missoula, MT",
        "gender" : "male"
};
```

JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C++, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras. En JSON, se presentan de estas formas:
Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con llave de apertura y termine con llave de cierre. Un arreglo es una colección de valores. Un arreglo comienza con corchete izquierdo y termina con corchete derecho. Los valores se separan por coma.[9]



Que es una sesión de usuario? La sesión se conoce como un intercambio de información interactiva semi-permanente,ya no esta constantemente activa, simétrica a un dialogo o una conversación, entre dos o más dispositivos de comunicación, entre un servidor y un usuario, que tiene algún interés en particular con este.[13]

El significado de Token. Un token, es un vale,que normalmente se le da a un usuario autorizado de un servicio computacional para facilitar el proceso de autenticación, y/o acceso.Existen dos tipos de token más usados, que son los siguientes:
Access token: Lleva contenida toda la información que necesita el servidor para saber si el usuario / dispositivo puede acceder al recurso que está solicitando o no. Suelen ser tokens caducos con un periodo de validez corto.[6]
Refresh token: El refresh token es usado para generar un nuevo access token. Típicamente, si el access token tiene fecha de expiración, una vez que caduca, el usuario tendría que autenticarse de nuevo para obtener un access token.[6]

Claim o reclamación. El conjunto de notificaciones JWT representa un objeto JSON cuyos miembros son los reclamos transmitidos por el JWT. Los nombres de reclamación dentro de un conjunto de notificaciones JWT tienen ser únicos. Los analizadores JWT deben rechazar JWT con nombres de reclamaciones duplicados o utilizar un analizador JSON que devuelve solo el último nombre de miembro duplicado léxicamente.Los reclamos o claims son declaraciones acerca de un objeto o entidad adicionandole cierto tipo de datos, como la expiración, el id,etc. Hay tres clases de nombres de reclamos de JWT: nombres de reclamos registrados, nombres de reclamos públicos y nombres de reclamos privados.[4]

1.2 Qué es JWT?

En esencia JWT es un bloque de información en formato JSON que esta firmado. Gracias a esta firma, se puede verificar su autenticidad, y al estar en un formato JSON, el tamaño es pequeño y familiar. Para una definición más rigurosa, la encuentra en el RFC 7519.[3] Ya con ciertos conceptos predefinidos, podemos usarlos para empezar a definir de forma más coloquial, el tema del que trata este articulo. JWT,son las siglas de Json Web Token,que es una de las herramientas más importantes en la parte de seguridad y autenticación que hace a todas las empresas que ofrezcan servicios por medios que requieran el uso de la Web. Es una herramienta sencilla, y lo suficientemente liviana, y esto es lo que lo hace tan factible su uso. Lo interesante de este articulo es que se queden con una idea de la estructura de un JWT, porque se usa, comparativa con los demás, y alguno detalles interesantes de esta tecnología, como aspectos negativos y positivos.

1.3 Términos asociados al tema.

Aunque la definición dada es un poco abstracta hasta ahora, no es difícil imaginar como se pueden utilizar: (aunque otros usos son posibles). Algunas de estas aplicaciones incluyen:
Autenticación:asegura que un usuario de un sitio web u otro servicio similar es auténtico o quien dice ser.
Autorización: un usuario que tenga facultad o derecho a solicitar o hacer algo.
Identidad federada:solución informática de gestión de las identidades, que puede ser interdependiente en los sistemas empresariales.
Sesiones del lado cliente (sesiones "sin estado")
Secretos del cliente.
Compactos:Debido a su tamaño más pequeño, los JWT pueden enviarse a través de una URL, un parámetro POST o dentro de una cabecera HTTP. Además, el tamaño más pequeño significa que la transmisión es rápida.
Autónomo:La carga útil contiene toda la información necesaria sobre el usuario, evitando la necesidad de consultar la base de datos más de una vez.

Inicio de sesión único(SSO):una vez que haces login, ya no es necesario hacerlo devuelta.
Aplicaciones de terceros:Soporte para token-api, ya que no todo usuario esta obligado del proceso de autenticación.
Claims:cierto numero de información del usuario/entidades empaquetadas en el JWT.
Autenticación via Token.[1, 5, 7]

1.4 Marco Teórico

Hoy en día existe una falta de seguridad, que es y determina uno de los factores importantes en un sistema seguro libre de contingencias, al asegurar los datos y los accesos estamos asegurando la continuidad de la operación de una organización reduciendo al mínimo los daños causados por una amenaza y manteniendo a salvo nuestros datos o patrimonios. Otro problema es asegurarnos de que la identidad digital sea la que se dice. Para poder tener una visión más clara sobre seguridad en la red, a continuación se realiza una breve descripción de los principios de información

Confidencialidad: Consiste en mantener la información secreta a todos, excepto a aquellos que tienen autorización para verla.

Integridad: Significa que se debe asegurar que la información no ha sido alterada por medios no autorizados o desconocidos.

Autenticidad: Debe ser posible para un usuario establecer el origen de la información, un atacante no debe tener la capacidad de hacerse pasar por otro usuario.Estas pautas son posibles tener en el JWT un aliado.

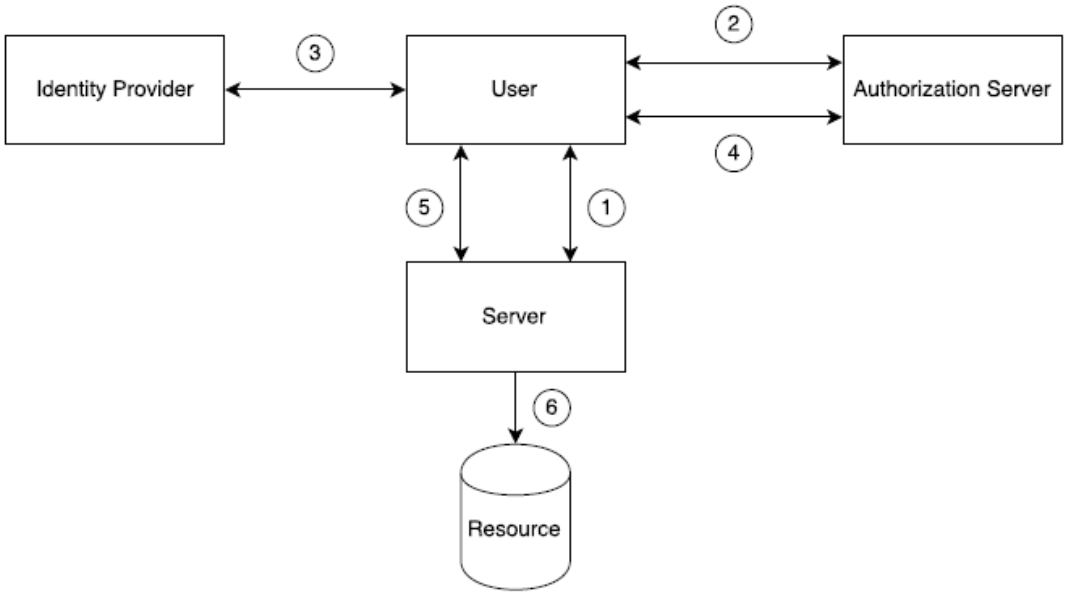
Aunque el propósito principal de los JWT es transferir información entre dos partes, el aspecto más importante de esto es el esfuerzo de normalización en una forma simple, opcionalmente validada y/o cifrada, en formato de contenedor de datos, que es lo que representa con la ayuda del JSON. Soluciones a este mismo problema se han implementado en privado y públicamente en el pasado. Existen también estándares más antiguos para establecer los reclamos sobre ciertas partes. Lo que JWT aporta a la mesa es un formato de contenedor simple, útil y estándar, esto por JSON como habíamos dicho, pero además los datos son firmados, aunque esto ya tenga un tiempo, por lo que la idea es mostrar como usar el formato para obtener servicios RESTful, sin iniciar sesiones, sin un estado que guardar de un usuario. Creo que una analogía del mundo real se entiende mejor. Digamos que usted quiera viajar de Paraguay a Uruguay, al cruzar de un país a otro, debe presentar alguna documentación propia de la Dirección General de Migraciones que avale su identidad.
Si usted presenta los papeles validos de su país, lo dejan pasar y a disfrutar de unas buenas y merecidas vacaciones en Punta del Este. Hagamos vuelta atrás y comparemos lo siguiente.
Dirección General de Migraciones - *servidor de autenticación que emitió el JWT.*
Documento que presento - *el JWT firmado por el servidor de autenticación, su identidad se dice que es legal por la firma.*
ciudadanía - *uno de los campos(claims) de su documento.*
Frontera - *capa de seguridad de la aplicación verificando su JWT y brindándole el acceso a los recursos, aunque mejor las vacaciones, no?*
Punta del Este - *recurso al que quiere acceder.* [3, 7]

1.5 Aplicaciones del JWT

Ahora podemos dar algunos ejemplos más prácticos donde podemos analizar la complejidad o simplicidad de un jwt, utilizado en las empresas de hoy día.

Lado del cliente, sesiones sin estado. Las llamadas sesiones sin estado son de hecho nada más que los datos del lado del cliente. El principal aspecto de esta aplicación recae en el uso de firmas y cifrados para auténticar y proteger los contenidos de la sesión. Los datos del lado clientes son siempre objetivos de robo, por lo tanto debe ser manejado con cuidado por el lado del backend. JWT puede proveer, varios tipos de firmas y cifrados. Las firmas son útiles para validar que el dato no fue manipulado. Y la encriptación es la encargada de proteger estos datos, para que no sean leídos por terceros. La mayoría de las sesiones sólo necesitan ser firmadas, en otras palabras, no hay seguridad ni privacidad en lo que concierne a los datos guardados en las sesiones, por lo tanto son fácilmente leídos por terceros. Un ejemplo muy común, es el claim sujeto que puede ser leído fácilmente, y es el encargado de identificar una de las partes con respecto a la otra. No tiene porque ser única, pero esto requiere como buena práctica tener más de un claims para identificar a un usuario sin error, pero esto queda a cargo de cada persona a la hora de su implementación.[7]

Identidad Federada. Los sistemas de identidad federados permiten a partes diferentes, posiblemente no relacionadas, compartir servicios de autenticación y autorización con otras partes. En otras palabras, la identidad de un usuario está centralizada. Existen varias soluciones para la administración de identidades federadas: SAML8 y OpenID Connect9 son dos de los más comunes. Algunas empresas ofrecen productos especializados que centralizan la autenticación y la autorización. Estos pueden implementar una de las normas mencionadas anteriormente o utilizar algo completamente diferente. Algunas de estas empresas utilizan JWTs para este propósito. El uso de JWT para la autenticación y autorización centralizadas varía de una empresa a otra, pero el flujo esencial del proceso de autorización se descrbie en la siguiente figura.[7]



1. El usuario solicita algún recurso al servidor. Necesidad de alguna autenticación.
2. Usuario, acorde al pedido del servidor, solicita un token de autenticación al servidor.Este tiene la configuracion de permitir al usuario login mediante una identidad federada.
3. El usuario hace el login con éxito con la autoridad federada
4. El usuario se vuelve a dirigir al servidor de autenticación, que usa las credenciales proveídas por la identidad federada
5. Ante el servidor de aplicaciones,la solicitud del usuario ahora tiene las credenciales correctas.
6. El usuario tiene el recurso solicitado.

2 Definición.Formato.Composición

JSON Web Tokens consta de tres partes separadas por puntos (.) Que son:
Header
Payload
Signature

Por lo tanto, un JWT normalmente se parece a lo siguiente. xxxxx.yyyyy.zzzzz

Vamos a desglosar las diferentes partes.

2.1 Cabecera.

El encabezado normalmente consta de dos partes: el tipo del token, que es JWT, y el algoritmo de hashing que se utiliza, como HMAC SHA256 o RSA. Cada JWT lleva una cabecera con claims sobre sí mismo. Estas afirmaciones establecen los algoritmos utilizados, si el JWT está firmado o encriptado, y en general, cómo analizar el resto del JWT. Según el tipo de JWT en cuestión, más campos pueden ser obligatorios en el encabezado. Por ejemplo, los JWT cifrados llevan información sobre los algoritmos criptográficos utilizados para el cifrado de claves y el cifrado de contenido. Estos campos no están presentes para JWTs no cifrados. El único reclamo obligatorio para una cabecera JWT no cifrada es el argumento **alg**: que es el algoritmo principal en uso para firmar y/o descifrar este JWT.

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

2.2 Carga útil.

La carga útil es el elemento donde normalmente se añaden todos los datos de usuario que son de interés, que son los claims. Los claims son declaraciones acerca de una entidad (normalmente, el usuario) y sus metadatos adicionales. Al igual que el encabezado, la carga útil es un objeto JSON. Ningún claim es obligatorio, aunque los claims definidos tienen un significado ya establecido, como las palabras reservadas en un lenguaje de programación. Los claims con significados especficos adjuntos a las mismas se conocen como claims registrados. De la carga útil, que contiene los claims, existen tres tipos de reclamos: reclamos reservados, públicos y privados.
Reserved claims: Se trata de un conjunto de claims predefinidos que no son obligatorios pero recomendados, para proporcionar un conjunto de claims útiles e interoperables. Algunos de ellos son: iss (emisor), exp (tiempo de caducidad), sub (sujeto), aud (público) y otros. Observe que los nombres de claims tienen sólo tres caracteres, ya que JWT está destinado a ser compacto.
Public claims: Estas pueden ser definidas a voluntad por aquellos que usan JWTs. Pero para evitar

colisiones deben ser definidos en el IANA JSON Web Token Registry o ser definidos como un URI que contiene un espacio de nombres resistente a colisiones.
Private claims: son las reclamaciones personalizadas creadas para compartir información entre las partes que acuerdan usarlas.

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

2.3 Firma.

Para crear la parte de la firma hay que tomar el encabezado codificado, la carga útil codificada, un secreto, el algoritmo especificado en el encabezado, y firmarlo.La firma se utiliza para verificar que el remitente de la JWT es quien dice que es y para asegurarse de que el mensaje no se ha cambiado en el camino.

2.4 JWT no seguros.

Con lo que vistos hasta ahora podemos ver un ejemplo más practico de un JWT. Este JWT no tiene ni firma ni cifrado. La cabecera tiene el claim **alg**, que es obligatorio, el valor *"none"*, que es lo que lo caracteriza como un JWT no seguro.Seria un JWT sin firma ni codificación En la práctica, sin embargo, los JWT sin garantía son raros.[1, 7]

2.5 JWT a partir de sus elementos.

Los algoritmos de generación en general, codifican en base64url la cabecera y la carga util. Luego los une, la cadena resultante de la codificación con un punto entre las mismas. A esta parte, cabecera más carga util, se le llama "datos".Luego con la firma JWT, y la cadena de "datos", se le realiza un hashing con la clave secreta usando el algoritmo de hash asignado en la cabecer del JWT.[12]

3 La necesidad del JWT

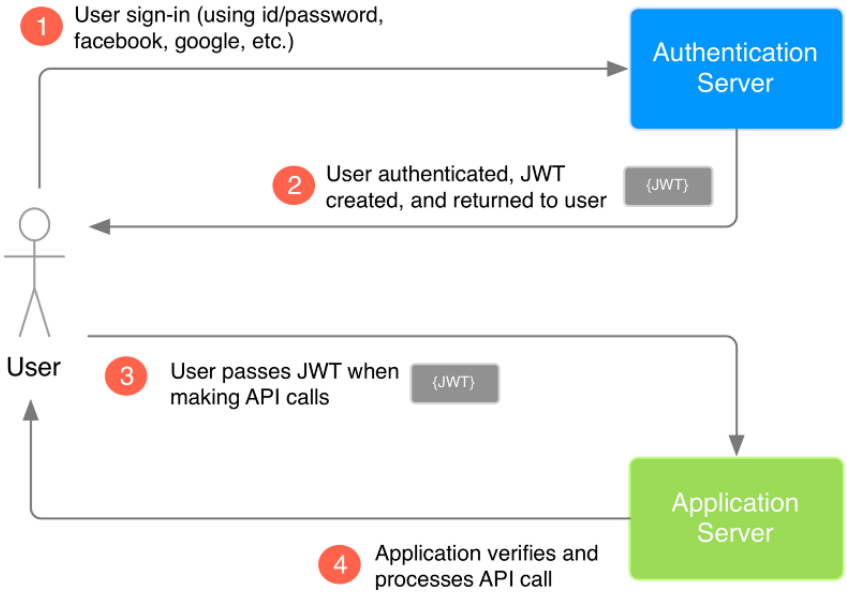
Durante mucho tiempo, la autenticación de usuarios en la web consistió en almacenar algunos datos muy simples (como un ID de usuario) en el navegador del usuario como una cookie. Esto funcionó bastante bien (y todavía lo hace para muchas aplicaciones), pero a veces se necesita más flexibilidad.Tradicionalmente, para obtener esta flexibilidad debía almacenar 'estado' en el servidor, lo que podría decirle cosas como quién es el usuario, qué tipo de permisos tienen, etc. Para almacenar estos datos, normalmente tenía que tener un archivo de datos dedicado/almacén, como Redis o una base de datos, que agregó a la complejidad de su aplicación. A partir de los últimos años, un nuevo estándar abierto ha venido que es cada vez más adoptado por algunos de los principales sitios web y aplicaciones. El estándar JWT. Con las cookies de sesión clásicas utilizadas en la mayoría de los sitios web, se almacena un token en el lado del cliente en una cookie del navegador (normalmente), que se utiliza para buscar los datos de sesión en el lado del servidor. Esto funciona bien para la mayoría de los sitios web sencillos, pero es necesario acomodar todos estos datos adicionales del lado del servidor con algún tipo de almacén de datos. Por otro lado, con JWTs puede guardar los datos de la sesión en el lado del cliente sin tener que preocuparse de que sea manipulado por el usuario u otro tercero, gracias a que está firmado por el servidor. Su cookie de sesión clásica también está firmada/codificada, pero al menos ahora los datos residen con el usuario. Almacenar el lado del cliente de datos también reduce algo de complejidad en el lado del servidor ya que ya no necesita un almacén de datos de baja latencia para almacenar y recuperar frecuentemente datos de sesión. Usted también consigue bastante más flexibilidad con el tipo de datos que almacena con JWTs. Dado que los datos están en el formato JSON, puede utilizar estructuras de datos profundamente anidadas. Claro, esto también es posible hacer con el lado del servidor de datos de sesión, pero de nuevo, con JWTs no es necesario tratar con otra base de datos. Por supuesto, usted puede tener mucha más información que esto en su ficha, pero esto suele ser un buen punto de partida para una nueva aplicación.[11]

3.1 Casos de usos convenientes.

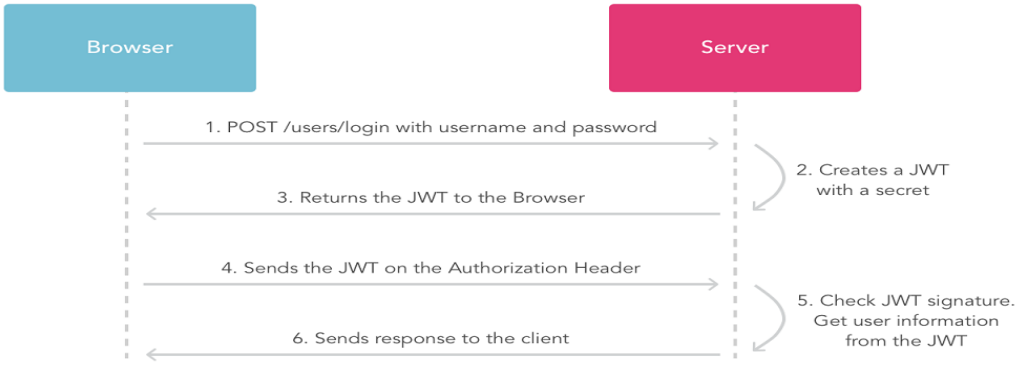
Aquí hay algunos escenarios donde JSON Web Tokens son útiles:

Autenticación : Este es el escenario más común para usar JWT. Una vez que el usuario ha iniciado sesión, cada solicitud posterior incluirá el JWT, permitiendo al usuario acceder a rutas, servicios y recursos que están permitidos con ese token. Single Sign On es una característica que utiliza ampliamente JWT hoy en día, debido a su pequeña sobrecarga y su capacidad para ser fácilmente utilizado en diferentes dominios.

Intercambio de información : Los ficheros web JSON son una buena forma de transmitir información de forma segura entre las partes. Debido a que los JWT se pueden firmar, por ejemplo, usando pares de claves públicas/privadas, puede estar seguro de que los remitentes son quienes dicen ser. Además, como la firma se calcula utilizando el encabezado y la carga útil, también puede verificar que el contenido no ha sido manipulado.[1]. Se puede ver como el usuario como primer paso lo que tiene que realizar es el proceso de login ante el servidor, una vez que el servidor compruebe que dicho usuario con sus credenciales correctas, y existen, se le envía un JWT con sus datos y claims necesarios. Cuando el usuario quiera hacer usos de los servicios que le provea vía

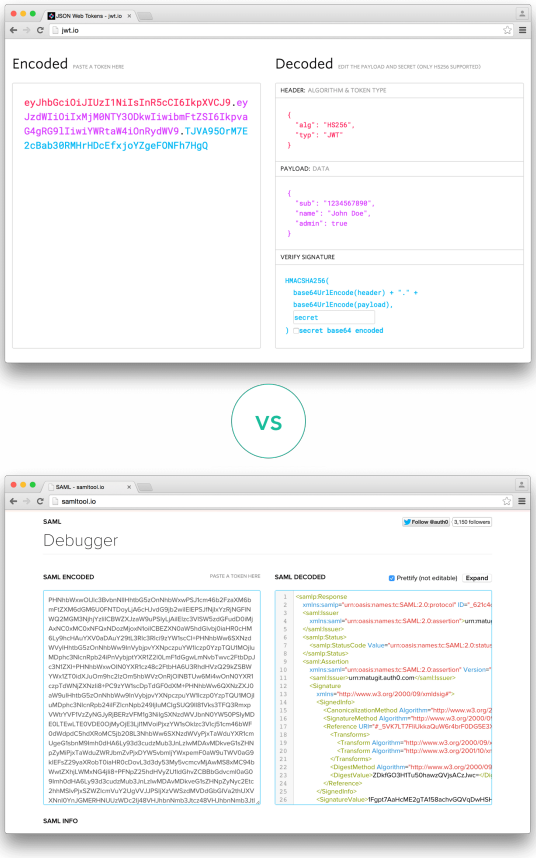


web la aplicación, debe enviarle el JWT que el servidor de autenticación le mando. El servidor de la aplicación luego comprueba que este JWT sea creado por el servidor de autenticación, y luego se asegura que la solicitud venga de un usuario registrado. Se trata de un mecanismo de autenticación sin estado, ya que el estado del usuario nunca se guarda en la memoria del servidor. Las rutas protegidas del servidor buscarán un JWT válido en el encabezado Autorización y, si está presente, se le permitirá al usuario acceder a los recursos protegidos. Como JWTs son autónomos, toda la información necesaria está ahí, reduciendo la necesidad de consultar la base de datos varias veces. Esto le permite confiar totalmente en API de datos que son apátridas e incluso hacer solicitudes a servicios de downstream. No importa qué dominios estén sirviendo sus APIs, por lo tanto, Cross-Origin Resource Sharing (CORS) no será un problema ya que no utiliza cookies.[1, 12].
El siguiente diagrama muestra este proceso.



4 Alternativas y comparaciones con otras tecnologías

Hablemos de las ventajas de JSON Web Tokens (JWT) en comparación con los Tokens Web Simple (SWT) y los Tokens de Lenguaje de Marcado de Asignación de Seguridad (SAML). Como JSON es menos detallado que XML, cuando se codifica, su tamaño es también más pequeño, haciendo JWT más compacto que SAML. Esto hace que JWT sea una buena opción para pasar en entornos HTML y HTTP. En cuanto a la seguridad, SWT sólo puede ser firmado simétricamente por un secreto compartido utilizando el algoritmo HMAC. Sin embargo, los tokens JWT y SAML pueden usar un par de claves públicas/privadas en forma de un certificado X.509 para firmar. La firma de XML con XML Digital Signature sin introducir oscuros agujeros de seguridad es muy difícil cuando se compara con la simplicidad de la firma de JSON. Los analizadores JSON son comunes en la mayoría de los lenguajes de programación porque se asignan directamente a los objetos. Por el contrario, XML no tiene una asignación de documento a objeto natural. Esto hace que sea más fácil trabajar con JWT que las aserciones de SAML. En cuanto al uso, JWT se utiliza en la escala de Internet. Esto pone de manifiesto la facilidad de procesamiento de cliente del token JSON Web en múltiples plataformas, especialmente móviles.[1].



4.1 Consideraciones para ambas opciones

Hay que tomar ciertas consideraciones para las opciones que estamos comparando.

En el caso de JWT.

1. Familia OAuth2 de especificaciones
2. Familia de especificaciones de OpenID Connect
3. JWE
4. JWS

Para SAML2, debemos considerar.

1. Familia de especificaciones SAML2
2. Firma digital XML
3. Cifrado XML
4. WS-Seguridad
5. WS-Trust

Teniendo en cuenta que es lo que queremos construir y en base a que familia de autenticación o en base a que cifrado o la estructura de datos que nos parezca más familiar o útil de usar, etc. Se notan mucho las variables que entran en juego a la hora de elegir que opción elegir, por eso queda a cada caso el análisis especifico. Pero dejamos una pequeña tabla bien resumida de las principales diferencias entre SAML2 y JWT.[2]

Variables	JWT	SAML2
Años	JWT RFC publicado en 2014	SAML2 introducido en 2005.
Adopción	OAuth2 y OpenID Connect.	Estándar de defacto para Enterprise SSO.
Filosofía	Sencillo.	Más enfoque académico. La sencillez no era una meta de diseño.
Estructura de datos	JSON	XML (estructura definida con XSD)
Tamaño	Mucho más pequeños a los de SAML	Tienden a ser muy grandes en comparación con JWT
Protocolos compatibles	API REST	Basado en SOAP
Protocolos de transporte	HTTPS	HTTP POST, HTTP GET, SOAP, JMS
Firmas digitales	JWS	Firma XML
Cifrado	JWE	CifradoXML
Infraestructura del lado del cliente	Construyendo una simple consulta HTTP	Generalmente requiere bibliotecas de soporte (pesado)

5 Cualidades del JWT

5.1 Sin almacenamiento de sesión

Ninguna sesión significa ningún almacenamiento de sesión. Suena como no mucho a menos que sus aplicaciones necesitan escalar horizontalmente. Si su aplicación se ejecuta en varios servidores, entonces compartir los datos de la sesión se convierte en una carga. Usted necesita un servidor especializado sólo para almacenamiento de sesión o espacio en disco compartido o sesiones pegajosas en el equilibrador de carga. Nada de eso es necesario cuando no se utilizan sesiones.[3]

5.2 Sin recolección de basura para las sesiones

Por lo general, las sesiones necesitan ser expiradas y la basura recogida. JWT puede llevar su propia fecha de caducidad junto con los datos del usuario. Por lo tanto, la capa de seguridad que comprueba la autenticidad de JWT también puede comprobar el tiempo de caducidad y simplemente rechazar el acceso.[3]

5.3 Servicios RESTful

Sin sesiones puedes crear servicios que sean verdaderos servicios RESTful,ya que se supone que el protocolo HTTP no contempla una estructura diferente(con estado).JWT es pequeño por lo que se puede enviar con cada solicitud, al igual que una cookie de sesión.Sin embargo, a diferencia de la cookie de sesión,que no apunta a ninguna información del servidor, JWT ya trae consigo los datos.[3]

6 Consideraciones de Seguridad

Existen casos de peligro a la hora de usar un JWT si no se tienen las consideraciones necesarias para implementarlas en cada caso. Por eso se dejan los siguientes casos que ya fueron considerados en experiencias pasadas para poder prever y no tener los mismos errores, y de esta forma sacarnos un peso de encima y tener menos problemas en nuestros proyectos. Iremos citando algunos de los ya considerados.[7]

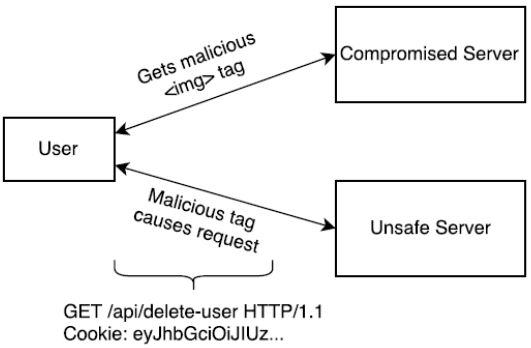
6.1 La eliminación de la firma.

Un método común para atacar un JWT firmado es simplemente eliminar la firma. Los JWTs firmados construidos a partir de tres partes diferentes que ya se mencionaron la inicio,el encabezado, la carga útil y la firma.El proceso de codificación del JWT es hacerlo en forma separada para las partes, de esta forma, se puede eliminar la firma y luego cambiar el encabezado para reclamar que el JWT no está firmado. El uso descuidado de ciertas bibliotecas de validación de JWT puede resultar en tokens no firmados que se toman como tokens válidos, lo que puede permitir al atacante modificar la carga útil a gusto. Esto se resuelve fácilmente asegurándose de que la aplicación que realiza la validación no considera válidos los JWT sin firmar.[7]

6.2 Falsificación de petición en sitios cruzados.

Estos ataques intentan realizar solicitudes contra sitios en los que el usuario inicia sesión engañando al navegador del usuario para que envíe una solicitud desde un sitio diferente. Para lograr esto, un

elemento especialmente diseñado debe contener la URL del destino. Un ejemplo común es una etiqueta `img` incrustada en una página malintencionada con la fuente apuntando a la pagina que es destino del ataque. Si el usuario estuviese registrado en la pagina y el sitio utilizó una cookie para mantener activa la sesión,la pagina que esta atacada,con este método,depende de si este sitio tiene o no alguna defensa contra CSRF, se tomara como una petición valida.Los JWT, como cualquier otro dato del lado del cliente, pueden almacenarse como cookies.[7]



6.3 Uso de Script entre sitios.

Se logran inyectando código JavaScript en sitios, donde el usuario tenga confianza, y puede robar tokens de acceso que todavía no expiraron. De esta forma un usuario malintencionado podría acceder a recursos protegidos.Estos ataques suelen ser causados por la falta de validación de los datos pasados al backend, similares a la inyección SQL. Por lo general con un comentario en una pagina, se logra inyectar el código, cada que que un usuario cargue la pagina, el código se ejecutara, y de esta forma robara las credenciales.[7]

7 Inconvenientes

No todo en este mundo tiene olor a rosas, y JWT no es la excepción. Por lo tanto, que nos podemos encontrar al usar JWT?

7.1 Al final...ocupa más espacio

Los tokens JWT no son exactamente pequeños. Especialmente cuando se usan sesiones sin estado, donde todos los datos están codificados directamente en el token, rápidamente superará el límite de tamaño de una cookie o URL. Es posible que decida almacenarlos en Almacenamiento local.[10]

7.2 Son menos seguros

Cuando guardas tu JWT en una cookie, ya no varia de cualquier otro identificador de sesión.Pero cuando guardas tu JWT en cualquier otro lado, ahora sos vulnerable a cualquier tipo de ataque.[10]

7.3 Invalidar tokens individuales

Y hay más problemas de seguridad. A diferencia de las sesiones - que pueden ser invalidadas por el servidor cada vez que se siente así - los tokens individuales JWT pueden ser invalidados. Por diseño, serán válidos hasta que expiren, pase lo que pase. Esto significa que no puede, por ejemplo, invalidar la sesión de un atacante después de detectar un compromiso. Tampoco puede invalidar sesiones antiguas cuando un usuario cambia su contraseña.[10]

7.4 Los datos se vuelven obsoletos

Esto puede significar que un token contiene alguna información obsoleta como una URL de sitio web antigua que alguien cambió en su perfil, pero más seriamente, también puede significar que alguien tiene un token con un rol de administrador, aunque acaba de revocar su función de administrador. Como tampoco puede invalidar los tokens, no hay manera de que elimine su acceso de administrador.[10]

7.5 Las implementaciones son poco probadas

Las implementaciones de sesiones existentes se han ejecutado en producción durante muchos, muchos años, y su seguridad se ha mejorado mucho por eso. No obtienes esos beneficios cuando usas tokens JWT, ya que muchas veces son implementaciones de terceros, sin mucho prueba. Lo cual seguramente introduzca problemas de seguridad a tu sistema.[10]

8 Conclusión

Es cierto que el JWT tiene sus issues o problemas, pero es muy versátil, y la forma de la implementación en nuestras empresas varia y pueden solucionarse muchas de ellas, mediante una implementación adecuada al caso. La autenticación que es lo fuerte del JWT, es cada vez más necesario, por lo que significa la flecha a donde se va el mercado. Los servicios web que puedan ser reutilizados en varias trabajos y necesiten tener escalabilidad, requiere tomas de medidas en el momento de la elección de las herramientas que uno va a utilizar, y por las características, JWT debe ser siempre tenida en consideración para estas cuestiones. Hay que tener en claro lo siguiente, el JWT es codificado como un objeto JSON, que una de sus características es la de su pequeño tamaño y fácil uso, esto permite aprovechar al máximo los recursos de la Web, quitar las trabas que podría tener la autenticación en las aplicaciones y bajar el consumo de recursos en los servidores de las mismas. Otra cosa muy positiva del JWT es la difusión que tuvo en el mercado, con cobertura en muchos lenguajes de programación, para sea cual sea el proyecto que queramos desarrollar el JWT sea una de las opciones que podamos tomar en el desarrollo. Cada vez más se tiene al JWT como forma de solucionar casos de autenticación, seguridad y autorización. Por la vivencia laboral que tuve hasta ahora, que es poca, pero aun así, JWT esta barajándose dentro de las posibles soluciones a aplicaciones que requieran alguno de estos procesos de autenticación, autorización. Por eso al menos el conocimiento de esta herramienta, aporta a nuestro futuro como profesionales en el área.

References

[1] Auth0. *Introduction to JSON Web Tokens*. URL: jwt.io/introduction/.

[2] Robert Broeckelmann. *SAML2 vs JWT: A Comparison*. 2016. URL: https://medium.com/@robert.broeckelmann/saml2-vs-jwt-a-comparison-254bafd98e6.

[3] Jacek Ciolek. *JSON Web Tokens (JWT) vs Sessions*. 2016. URL: float-middle.com/json-web-tokens-jwt-vs-sessions/.

[4] IETF. *JSON Web Token (JWT)*. URL: https://tools.ietf.org/html/rfc7519.

[5] OKONKWO VINCENT IKEM. *How We Solved Authentication and Authorization in Our Microservice Architecture*. 2017. URL: initiate.andela.com/how-we-solved-authentication-and-authorization-in-our-microservice-architecture-994539d1b6e6.

[6] Laura Celeste Diaz Villar y Jorge Carrillo Villalba. "Vulnerabilidades sobre mecanismos de seguridad basadas en metodos de autenticacion Json Web Token." In: (2017).

[7] *JWT HandBook*. Auth0 Inc, 2016.

[8] Jason Lengstorf. *JSON: What It Is, How It Works, How to Use It*. 2014. URL: www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/.

[9] Json Organization. *JSON*. URL: www.json.org/json-es.html.

[10] Joe Pie. *Stop using JWT for sessions*. 2016. URL: cryto.net/~joepie91/blog/2016/06/13/stop-using-jwt-for-sessions/.

[11] Scott Robinson. *Understanding JSON Web Tokens (JWT)*. 2015. URL: stackabuse.com/understanding-json-web-tokens-jwt/.

[12] Mikey Stecky-Efanti. *5 Easy Steps to Understanding JSON Web Tokens (JWT)*. 2016. URL: medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec.

[13] Wikipedia. *Sesion (informatica)*. URL: https://es.wikipedia.org/wiki/Sesion.