19 de febrero de 2015

¿Qué es la autenticación basada en Token?

Existen varios sistemas de autenticación en una aplicación web. A continuación veremos las 2 versiones más utilizadas junto con sus ventajas e inconvenientes.

Autenticación en el servidor, almacenando la sesión

El más común hasta ahora era el que guardaba en una sesión la información del usuario. Para ello necesitábamos almacenar esa información en una base de datos, podía ser una colección de **MongoDB** o en **Redis**.

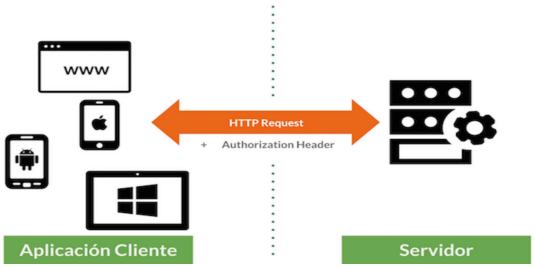
Sin embargo esto suponía una pérdida de escalabilidad en nuestra aplicación, ya que el servidor debe almacenar un registro por cada vez que el usuario se autentique en el sistema. Además hacemos que el Backend se encargue de ello y de esta manera si queremos desarrollar una aplicación móvil, necesitaríamos otro backend diferente, no pudiendo reutilizarlo.

Autenticación sin estado con Tokens

Por ello una de las nuevas tendencias en cuanto al desarrollo web moderno se refiere, es la autenticación por medio de *Tokens* y que nuestro backend sea un API RESTful sin información de estado, *stateless*.

El funcionamiento es el siguiente. El usuario se autentica en nuestra aplicación, bien con un par usuario/contraseña, o a través de un proveedor como puede ser Twitter, Facebook o Google por ejemplo. A partir de entonces, cada petición HTTP que haga el usuario va acompañada de un *Token* en la cabecera. Este Token no es más que una firma cifrada que permite a nuestro API identificar al usuario. Pero este Token no se almacena en el servidor, si no en el lado del cliente (por ejemplo en *localStorage* o *sessionStorage*) y el API es el que se encarga de descrifrar ese Token y redirigir el flujo de la aplicación en un sentido u otro.

Como los **tokens son almacenados en el lado del cliente**, no hay información de estado y la aplicación se vuelve totalmente escalable. Podemos usar el mismo API para diferentes aplicaciones (Web, Mobile, Android, iOS, ...) solo debemos preocuparnos de enviar los datos en formato JSON y generar y descrifrar tokens en la autenticación y posteriores peticiones HTTP a través de un middleware.



También nos **añade más seguridad**. Al no utilizar cookies para almacenar la información del usuario, podemos evitar ataques CSRF (*Cross-Site Request Forgery*) que manipulen la sesión que se envía al backend. Por supuesto podemos hacer que el token expire después de un tiempo lo que le añade una capa extra de seguridad.

Autenticación con JSON Web Tokens

El estándar para este tipo de autenticación es utilizar JSON Web Tokens (JWT). Al igual que los APIs, el formato JSON es agnóstico del lenguaje, y podemos utilizar el que queramos (Node.js, Python, Ruby, PHP, .NET, Java,...)

El formato de un **JWT** está compuesto por 3 `strings` separados por un punto `.` algo así como:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1NGE4Y2U2MThlOTFiMGIxMzY2NWUyZjkiLCJpYXQiOiIxNDI0MTgwNDg0IiwiZXhwIjoiMTQyNTM5MDE0MiJ9.yk4nouUteW54FlHbWtgg1wJxeDjqDA_8AhUPyjE5K0U
```

Cada `string` significa una cosa:

- **Header** La primera parte es la cabecera del token, que a su vez tiene otras dos partes, el tipo, en este caso un JWT y la codificación utilizada. Comunmente es el algoritmo *HMAC SHA256*. El contenido sin codificar es el siguiente:

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

Codificado sería: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9`

- **Payload** EL *Payload* está compuesto por los llamados *JWT Claims* donde irán colocados la atributos que definen nuestro token. Exiten varios que puedes consultar aquí, los más comunes a utilizar son:
  - `sub` : Identifica el sujeto del token, por ejemplo un identificador de usuario.
  - `iat` : Identifica la fecha de creación del token, válido para si queremos ponerle una fecha de caducidad. En formato de tiempo UNIX

- `exp` : Identifica a la fecha de expiración del token. Podemos calcularla a partir del `iat`. También en formato de tiempo UNIX.

```
{
   "sub": "54a8ce618e91b0b13665e2f9",
   "iat": "1424180484",
   "exp": "1425390142"
}
```

También podemos añadirle más campos, incluso personalizados, como pueden ser el rol del usuario, etc.

```
{
   "sub": "54a8ce618e91b0b13665e2f9",
   "iat": "1424180484",
   "exp": "1425390142",
   "admin": true,
   "role": 1
}
```

Codificado sería:

```
eyJzdWIiOiI1NGE4Y2U2MThlOTFiMGIxMzY2NWUyZjkiLCJpYXQiOiIxNDI0MTgwNDg0IiwiZXhwIjoiMTQyNTM5MDE0MiJ9
```

- **Signature** La firma es la tercera y última parte del JSON Web Token. Está formada por los anteriores componentes (Header y Payload) cifrados en *Base64* con una clave secreta (almacenada en nuestro backend). Así sirve de *Hash* para comprobar que todo está bien.

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

Codificado sería: `yk4nouUteW54F1HbWtgg1wJxeDjqDA_8AhUPyjE5K0U`

Por tanto, todo nuestro JSON Web Token, una vez codificado tendrá esta pinta:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1NGE4Y2U2MThlOTFiMGIxMzY2NWUyZjkiLCJpYXQiOiIxNDI0MTgwNDg0IiwiZXhwIjoiMTQyNTM5MDE0MiJ9.yk4nouUteW54F1HbWtgg1wJxeDjqDA_8AhUPyjE5K0U
```

Que si lo comprobamos en la web JWT.io vemos que nos lo traduce a los campos que hemos visto.

En la siguiente entrada vemos como implementar **este tipo de autenticación en Node.js** y más adelante en el **lado cliente con AngularJS**.

Si te ha gustado éste artículo, te animo a compartirlo para que otros puedan leerlo :)

Carlos Azaustre

http://carlosazaustre.es

CTO y Co-fundador en Chefly. Desarrollador web full-stack. Formador y evangelizador en tecnologías web, sobre todo en el ecosistema JavaScript. Es autor del ebook Aprendiendo JavaScript y del curso online Fundamentos de React.js. También puedes seguir sus videotutoriales y vlogs en su canal de Youtube

46 Comentarios Carlos Azaustre / Front End Developer Edwin Ramiro Os...

Recomendar 4 Compartir Ordenar por los mejores

Carlos Azaustre / Front End Developer requiere que verifiques tu email antes de publicar. Enviar correo electrónico de verificación a edwin eka@hotmail.com

Únete a la conversación...

jer • hace un año

Por cierto, hay un problema de seguridad crítico al usar JWT, cuando se determina el algoritmo a usar para generar la asignatura (en el campo alg en la cabecera), aun no se ha validado el token por lo que no se ha validado la cabecera del token en sí.
En orden de validación del token, le permitimos al atacante que metodo usar para verificar la asignatura.

Buscando una solución, acá una serie de recomendaciones al respecto
https://auth0.com/blog/2015/03...
1 • Responder • Compartir ›

Carlos Azaustre Moderador jer • hace un año

Gracias por tu aporte @Jer!
• Responder • Compartir ›

Jose Guadalupe Rodriguez Flore • hace 3 meses

Hola , Saludos, tengo una duda, ¿se podria utilizar esto para enviar y recibir datos desde un Ajax a un Webservices? o ahi que podria aplicar para que los datos no se vean?
• Responder • Compartir ›

Wilmer Márquez • hace 3 meses

Hola carlos, tengo una pregunta...

Si utilizo el metodo de autenticacion por JWT para por ejemplo autenticar una app para movil con un token que dure por ejemplo, 15 dias... si el usuario pierde el movil, hay alguna manera de revocar esa autorizacion.?? ya que si el usuario cambia la contraseña desde la pc el token seguiria siendo valido desde el movil

Responder • Compartir ›

sk8sk8 • hace 6 meses

Hola Carlos,

Muchas gracias por compartir esta información. Estoy plagado de dudas y tal vez puedas darme algo de luz. Voy a empezar con algunas a ver que tal.

1. Entiendo que el flujo para conseguir este token sería pasando por un login con, por ejemplo, basic auth. Correcto?
2. Como se revoca un token? Imagino la típica situación de cambio de contraseña donde los tokens generados no sirven ya (tal vez por que piensa que le robaron la contraseña).
3. Entiendo que esto no está sujeto a CSRF, cual es la forma correcta de almacenar este token?
4. Hay alguna forma de renovar la caducidad del token a cada llamada sin generar uno nuevo? Imagino el caso de poder poner una caducidad de 10 minutos y si no hace ninguna llamada en ese tiempo caduca, pero si la hace va renovándose, para controlar inactividad (¿o tal vez esto se controla desde el cliente?)
5. Que opinas de que uno de los parametros personalizados sea la IP y comprobarla a cada llamada posterior con la actual del usuario por seguirdad?
7. Siempre se ha dicho que tener un salt (o secret) para cifrar las contraseñas no es seguro, ya que si te obtienen el secret podrían obtener las password, por ello se usa bcrypt. En cambio veo que JWT necesita de un secret, ¿Sería entonces inseguro? ¿Crees viable algun tipo de mezcla de bcrypt con jwt con el objetivo de evitar este secret? (he visto que el payload puede obtenerse sin el secret en jwt.io con lo que tal vez el payload salteado y hasheado podría ser el secret ¿?)

Muchas gracias por echarme un cable!

Responder • Compartir ›

sk8sk8 ➔ sk8sk8 • hace 6 meses

Y es que, por mucho que le doy vueltas y viniendo del mundo "no escalable" de las sesiones almacenadas en un redis, me choca mucho que los tokens sean validos incluso cuando se ha hecho un logout sobre ese token.

Responder • Compartir ›

sk8sk8 ➔ sk8sk8 • hace 6 meses

Interesante opinión sobre las revocaciones: https://www.dinochiesa.net/?p=...

Responder • Compartir ›

Eduardo • hace 7 meses

Hola Carlos, sabes como hacer para que no expire ? yo uso auth0 !!! con angular.
Saludos
Eduardo

Responder • Compartir ›

yosimar • hace 7 meses

Hola Carlos Azaustre interesante tu articulo ahora como hago para leer un token que me manda otra aplicacion y necesito leerlo y acceder a mi aplicacion sin necesidad de loguearme

Responder • Compartir ›

Lixander Ricardo Rodríguez • hace 7 meses

a ver, una pregunta, tengo un blog, pero no me interesa que los usuarios publiquen noticias en el ni nada, este tipo de autenticación no me serviría de nada no?

Responder • Compartir ›

Carlos Azaustre Moderador ➔ Lixander Ricardo Rodríguez • hace 7 meses

Hola **@Lixander Ricardo Rodríguez**, Este tipo de autenticación es como cualquier otra, la diferencia de los tokens es que te permite no tener que almacenar sesiones. Sirve mucho si tienes una aplicación con un API REST que es accesible desde varias app: (web, ios, android, etc...) En el caso de un blog, el servidor tiene el acceso a la BD y renderiza el contenido. Entonces no es necesario autenticación más que el login del admin. y los comentarios puedes utilizar Disqus como utilizo en mi blog :)

Un saludo!

Responder • Compartir ›

Lixander Ricardo Rodríguez ➔ Carlos Azaustre • hace 7 meses

ohhh que interesante gracias **@Carlos Azaustre**, muy buena la información, de todas formas, si desea puede visitar mi blog www.tunel.cult.cu , es el blog de la cultura en mi provincia Las Tunas acá en Cuba, acepto opiniones en los comentarios del mismo, malas o buenas, ambas me sirven, un saludo, muy bueno el blog :)

Responder • Compartir ›

Martin Castro • hace 8 meses

Hola Carlos.

Muy bien explicado!!

Un saludo desde Argentina

Responder • Compartir ›

Angel Uc • hace 8 meses

Hola, espero puedas responderme, ya entendí como Crear el Token, pero ahora cual es el siguiente paso? me refiero a mis methodos rest api, que recibiran? 2 parametros? uno el token y 2 el json (el cual este json puede contener el/los objetos con el que quiero trabajar) o tendra 1 json? donde este json contendra el token y a su vez el/los objeto(s) con los que quiero trabajar? o bien esto ya depende de como yo lo quiera poner? o todo lo que mande tiene que estar "encriptado" con el token? (en su caso como se haria eso?) bien ahora hablando de lado del server suponiendo que ya recibir el objeto y token, que debo hacer con el token? debo validar que el usuario existe (debidamente debi guardar en los "claim" quiza el id del usuario y contraseña o algo asi) tambien validar que no halla expirado (o esto lo hace automaticamente?) que cosas debo hacer con el key? gracias.

Responder • Compartir ›

Carlos Azaustre Moderador ➔ Angel Uc • hace 8 meses

Hola **@Angel Uc**,
En tus peticiones desde el cliente, enviarás un JSON como cuerpo del mensaje, pero el token viaje en los HEADERS. En tu API debes comprobar si la petición que le llega, contiene en sus HEADERS el token, comprobar si es un token valido y que sea de un usuario registrado, y despues ya tratar el cuerpo del JSON.

Espero que te sirva.

Un saludo!

1 Responder • Compartir ›

leandro • hace 9 meses

Hola, tengo unas preguntas tal vez tontas, si en el servidor no se almacena ni siquiera el token, que pasa si alguien llegara a obtener el token y lo usara desde otro dispositivo? Que pasa si cierro sesión en mi dispositivo y alguien que tenga el token lo usa antes de que expire? Que pasa si cambio de contraseña en uno de mis dispositivos y en otro intento usar el token que ya tenia, pues como el token no contiene la contraseña, como valida el servidor que ese token ya no es válido?

Responder • Compartir ›

Axel Fernando Gallegos ➔ leandro • hace 8 meses

Hola leandro, primero tus preguntas no son tontas tiene una validez cuando hablamos de seguridad y hay que tenerla en cuenta, creo que para resolver varios de los problemas planteados es que tu hagas un seguimiento de los token que has creado, recuerda que el token se creara cuando alguien inicie en tu aplicación, en ese momento recolecta mas datos y almacena los permisos que le has dado a tu usuario ademas del token así podrás hacer un seguimiento de las solicitudes y los dispositivos en que se hace la solicitud, recuerdas que entre mas seguridad tengas mejor sera tu sistema,con lo del cambio de contraseña lo que puedes hacer es habilitar o deshabilitar el token en tus registro así el token aunque no este caducado estará deshabilitado para tu sistema, espero haya podido aclarar tus dudas.

Responder • Compartir ›

Wilmer Marquez ➔ Axel Fernando Gallegos • hace 3 meses

Axel una pregunta, al hacer un registro de todos los tokens generado no estoy haciendo una mala practica y rompiendo el principio de los JWT que es una autenticacion sin estado que no se almacena en el servidor.???

Responder • Compartir ›

Santiago Rodríguez • hace 10 meses

Carlos cual seria la utilidad de tener un json con información como Header y el Payload, si con el token generado, haciendo una comparación con el token que tenga en el servidor puedo obtener la información del usuario, así tambien me evito volver a genera el token para validar si la información que va en el json no es corrupta

Responder • Compartir ›

Carlos Azaustre Moderador ➔ Santiago Rodríguez • hace 8 meses

Hola **@Santiago Rodríguez**, el token no es JSON, es una cadena de texto que viaja en el HEADER. Como dices, luego en el servidor obtienes ahí la información :)

Espero que te sirva. Saludos!

Responder • Compartir ›

Sebastian Vega • hace un año

hola, queria consultar como podria utilizar esto desde un cliente a un servicio PHP para añadir seguridad (limitar el acceso a los servicios). La idea que tengo en la cabeza es que el cliente debe enviar los datos encriptados con la "key" y desde el lado del servidor desencriptar estos datos con la misma "key", y si coinciden dejarlo utilizar el servicio. El tema es que no se aun como realizar la comparacion al desencriptar el dato.

Si me pudieses brindar algun tipo de ayuda te lo agradezco mucho

Responder • Compartir ›

Axel Fernando Gallegos ➔ Sebastian Vega • hace 8 meses

Hola **@Sebastian Vega**, creo que lo que quieres hacer tendría fallas en lo que se trata de seguridad, lo recomendable es que primero el usuario inicie sesión en el servidor y si es validado se devuelva el JWT con las validaciones de los servicios, así el usuario no sabe cual es la clave, solo la sabrá el servidor, ya que hablamos de php te recomiendo que utilices slim te permite crear un webservice y trae una función para jwt, prueba con ello.

Responder · Compartir ›

Carlos Azaustre Moderador → Sebastian Vega • hace 10 meses
Hola **@Sebastian Vega**, muchas gracias por comentar.

Con PHP no tengo mucha experiencia. Yo buscaría si existe alguna librería para el framework que estés utilizando que realice algo parecido a lo que buscas. Lo ideal es lo que comentas, que los datos estén encriptados con una clave que solo el servidor conoce para que solo sea el Backend el que lo pueda leer.

Un saludo!
Responder · Compartir ›

fran25 • hace un año
Hola Carlos. Enhorabuena por el Post!!!

Respecto al post tengo varias dudas en lo que respecta a seguridad (Autenticación con Tokens). El caso es que estoy leyendo en los últimos días que almacenar el token en el WebStorage (localStorage/sessionStorage) es inseguro, ya que estamos expuestos a ataques XSS. Se recomienda, por tanto, almacenar el token en una cookie (con httponly, ssl...). La cuestión es: ¿y si evitamos en la medida de lo posible almacenar el token?. Por ejemplo, en una aplicación con Angular, lo guardamos de forma global (por ejemplo en una factoría) y ya tendríamos acceso al token sin guardarlo ni en cookies ni en webstorage. El único problema que presenta es en el caso de refrescar la página, que se pierden los valores de todas las variables. Sólo en este caso, lo que haríamos es, antes de perder los valores (por ejemplo en el evento beforeunload) guardamos en localstorage el token y cuando termina de cargarse la página cogemos ese valor, lo guardamos en la variable global y eliminamos el valor de localstorage. ¿Esto sería más seguro o estamos añadiendo complejidad innecesaria a la app de angular?

Muchas gracias!!
Responder · Compartir ›

Carlos Azaustre Moderador → fran25 • hace un año
Hola @fran25, gracias por comentar.

Sobre la seguridad, tienes la misma que una cookie, ya que ambas se almacenan en el navegador y van codificadas.

Con el token te ahorras tener que mantener una sesión, y desacoplar el Backend del frontend.

Otras alternativas a Cookies o Token, me parecen que complican el desarrollo como dices :)

Un saludo!
Responder · Compartir ›

John Mauricio Carmona Escobar → Carlos Azaustre • hace un año
Hola Carlos, primero que todo muchas gracias.

El inconveniente que veo es que cualquier persona puede copiar la cookie que almacena la sesión y pegarla en otra parte y estaría autenticado.

Que se podría hacer en esos casos?
Responder · Compartir ›

Carlos Azaustre Moderador → John Mauricio Carmona Escobar • hace un año
El token o cookie se almacena en tu navegador para el dominio en el que te encuentras. La única forma de copiar ese token es entrando físicamente en tu ordenador, visitar la página estando logueado y copiandola manualmente del localStorage.

Desde otro dominio no pueden acceder al localStorage de otro dominio.

Espero que te sirva
Responder · Compartir ›

John Mauricio Carmona Escobar → Carlos Azaustre • hace 9 meses
Carlos muchas gracias
Responder · Compartir ›

Axel Fernando Gallegos → John Mauricio Carmona Escobar • hace 8 meses
@John Mauricio Carmona Escobar te recomiendo en ves de utilizar localStorage utiliza SessionStorage eso te almacenara por sesión, lo otro es hacer un seguimiento al token eso te dará mas seguridad en tu aplicación
Responder · Compartir ›

John Mauricio Carmona Escobar → Axel Fernando Gallegos • hace 8 meses
@Axel Fernando Gallegos Muchas gracias.
Aun que no me quedo muy claro lo de hacer un seguimiento al token pero lo voy a investigar. De nuevo muchas gracias
Responder · Compartir ›

Alejandro Ventura • hace un año
Excelente explicación Carlos!
Mis pregunta con:
- ¿cómo podría forzar la finalización de la sesión actual del usuario, es decir que haya una opción de "Logout" para no tener que esperar a que el token?
- Entonces si el usuario esta contestando un formulario y el token caduca ¿Debo forzar al usuario a regresar a la pagina de login para firmarse otra vez?

Espero puedas ayudarme, te mando un saludo!
Responder · Compartir ›

Carlos Azaustre Moderador → Alejandro Ventura • hace un año
Hola **@Alejandro Ventura**, gracias por comentar.

Respecto a tus preguntas, para finalización del token, con tener una función que lo que haga sea eliminar el token del sessionstorage o localstorage te sería suficiente. Despues también puedes redirigir a otra URL para que se actualice la página.

Y si el token caduca, cuando el usuario vaya a realizar una acción en la que sea necesario estar autenticado, si lo tienes bien implementado te redirigirá a la página de login.

Ten en cuenta que estos tokens pueden tener perfectamente un tiempo de vida de 15 días o los que tu configures es tu backend. En las webs de Banca suele ser más común que expiren en 5 minutos por seguridad, pero por ejemplo en Facebook o Twitter, si no te deslogueas siempre puedes entrar directamente a tu cuenta.

Espero que te sirva. Un saludo!
Responder · Compartir ›

Alejandro Ventura → Carlos Azaustre • hace un año
Excelente, me ha servido tu respuesta. Solo que no se como eliminar el token del sessionstorage o del localstorage. Tienes algún articulo que me pueda ayudar? Te agradecería bastante Carlos.
Saludos y muchas gracias por contestar!
Responder · Compartir ›

Carlos Azaustre Moderador → Alejandro Ventura • hace un año
Para eliminarlo únicamente has de borrarlo del localstorage. Si al item lo llamas "token" por ejemplo, se puede eliminar con el método: localStorage.removeItem('token')

Saludos!
Responder · Compartir ›

José Muñoz Ruiz • hace un año
Carlos, unas pequeñas grandes dudas...

1.- El usuario se autentica, ya sea por user/password, Facebook,Twitter, etc.

2.- Pos validación, se le entrega al cliente un TOKEN que se puede guardar del lado del cliente,por ejemplo en SessionStorage.

3.- (acá mi duda)
-Quien es el que genera el formato JWT?
-Se genera por el lado del cliente o lo genera el API en su lógica ?
-Si se genera por el lado del cliente de que manera se envía si consumo el API por ajax. Por parametro URL (GET)?
-Que se envía exactamente solo el token? o todo el formato JWT?
- Si se envía desde el lado del cliente todo el formato JWT, el API el debe poder decodificar esto para validar que esta todo OK??
Responder · Compartir ›

Carlos Azaustre Moderador → José Muñoz Ruiz • hace un año
Hola **@José Muñoz Ruiz**, muchas gracias por plantear tu duda.

El Token o JWT lo genera el servidor api con una clave secreta que tiene el servidor y los datos del usuario que se quieran codificar.

Se envía en las cabeceras de la petición. Se puede ver en la consola del navegador si accedemos a la pesta˜ã de Red y clickamos en una petición concreta. Se envía así tanto desde el cliente como desde el servidor

El Api es el que se encarga de decodificarlo, ya que es el único que conoce el "secret" y ya despues realizas la lógica que necesites :)

Saludos!
Responder · Compartir ›

Angel Uc → Carlos Azaustre • hace 8 meses
Yo tengo dudas similares, estoy trabajando con c#, que es lo que voy a recibir en el servidor? un json? por ejemplo para crear un "post" que contiene un titulo, descripcion, le envio el json y dentro del json contendria mi objeto "post" y mi "objeto" "token"?
Responder · Compartir ›

Axel Fernando Gallegos → Angel Uc • hace 8 meses

hola **@Angel Uc** tienes que enviar el token desde la cabecera HEADER, los otros datos los envías desde post o el método que utilices.
El servidor tomara el header y hay tendras que agregar esto Authorization: Bearer <token> donde se hara la validación
1 ^ v • Responder • Compartir ›

Pablo Rodriguez • hace un año
Hola Carlos tengo dudas con tu comentario(El Secret está almacenado en tu servidor)

Esta muy claro que no se deben colocar los tokens oauth en nuestras apps.

Pero si nuestra app que estara instalada en muchos dispositivos y queremos que pueda hacer a cosas en una cuenta por medio de un Token oauth.

Donde le doy al "usuario" esa llave para que acceda a direferentes requests desde una app en iOS x ejemplo hacia nuestra Web API.

He probado el esquema HMAC-API Key Authentication, pero compartiendo el API key desde una aplicación cliente.
En este caso como puedo hacer para diferenciar un Request Valido de uno Malicioso por ejemplo.
Si no se puede compartir un una llave de acceso previa en nuestra app?
^ v • Responder • Compartir ›

Axel Fernando Gallegos ➔ Pablo Rodriguez • hace 8 meses
Creo que tu pregunta no va con el post ya que utilizas oauth, pero si hablamos de JWT, las validaciones se hacen en el servidor no en el cliente, tu puedes almacenar el token en tu app, pero tu palabra secreta la almacenas en tu servidor
^ v • Responder • Compartir ›

diferrev • hace 2 años
Hola! o,O
Es hora de empezar con autentificación por medio de Tokens.

Gran post!
^ v • Responder • Compartir ›

Lugo • hace 2 años
Hola Carlos, excelente post!.

Una consulta amigo, tengo problemas al generar el signature, estoy intentando con la libreria CryptoJS, pero no logro que me quede igual al de tu ejemplo. Podrías recomendarme alguna libreria para encryptar en HMACSHA256. Estoy intentando hacerlo sin usar libreria de token(Ya vi que recomiendas una.)

En cuanto al secret que se genera con el que se encrypta.

Es una llave "FIJA" que nunca cambia? Estoy usando Cordova Phonegab para hacer una app para IOS, en este caso la clave "Secret" la tengo que poner tanto en el código de la app como en el del servidor dentro de una variable Fija que nunca cambiara? o cual seria la estrategia para esta clave.

Gracias por tu apoyo!
^ v • Responder • Compartir ›

Carlos Azaustre Moderador ➔ Lugo • hace 2 años
El Secret está almacenado en tu servidor, es con el que se generan los Tokens y digamos que es como la "llave" que abre esos tokens para poderlos leer.

Puedes cambiarla en tu servidor de vez en cuando, es recomendable. Digamos que es como cuando cambias la contraseña de tu email por seguridad. Debe estar en el servidor y si es posible dentro de las variables de entorno del sistema para que no pueda ser accedida facilmente. No la pongas en tu app cliente.

Para encriptarlo y generarlo lo hago en el servidor y uso la librería de NodeJS 'jwt-simple' y el método .encode() al que solo le paso el Payload y el se encarga de generar la firma y crear el token.
^ v • Responder • Compartir ›

Miguel Angel Martin Hernandez • hace 2 años
Hola Carlos.

Antes de nada aprovecho para felicitarte por el estupendo trabajo que haces por y para la comunidad, eres un referente muy importante y tu blog es una lectura must-have.

Respecto al post sólo tengo una duda y es la siguiente. Cuando dices que Signature está compuesto por Header y Payload cifrados con un secret en Base 64 y que así sirve como Hash ¿A qué te refieres exactamente con Hash? ¿Por si te modifican Payload y poder comparar todo?

No se si consigo explicarme. :)
^ v • Responder • Compartir ›

Carlos Azaustre Moderador ➔ Miguel Angel Martin Hernandez • hace 2 años
Hola **@Miguel Angel Martin Hernandez**, muchas gracias por seguirme y comentar :)

Si eso es es. Es un código que confirma que todo lo anterior es correcto, como una validación de que ningún digito del token ha sido cambiado.

Un saludo!
^ v • Responder • Compartir ›

Jairo Jorquera ➔ Carlos Azaustre • hace un año
Hola, por casualidad leiste esto: https://auth0.com/blog/2015/03...
^ v • Responder • Compartir ›