Tema 4: Mecanismos de autenticación, autorización y control de acceso

Los servicios REST/JSON gestionan eventos corporativos. Los conceptos que se estuduan son los aplicables a otro tipo de servicios como SOAP, Thrift, gRPC, etc… Un servicio REST es un servicio que implementa la lógica de negocio, pero que no tiene interfaz gráfica y es utilizado por un servicio o aplicación que si tiene interfaz gráfica. Un ejemplo de esto sería una aplicación para ver el tiempo en el móvil, la aplicación contactaría con un servicio REST por HTTP para solicitar la información climatológica y el REST devolvería la información en un formato estructurado como por ejemplo XML o JSON. Este mismo servicio rest podría ser utilizado por otras aplicaciones con una interfaz gráfica diferente.

Vamos a ver un ejemplo base. Normalmente dentro de estos servicios suelen haber roles como:

- User
- Manager

El servicio atiende a peticiones HTTP para:

- [Manager] Crear un evento
- [Manager] Manager
- [No autentciado] Buscar eventos
- [No autenticado]Obtener información de un evento
- [User]responder a un evento
- [User y Manager]Recuperar las respuestas de un usuario

Dependiendo de lo que se vaya viendo en cada apartado se analizará este ejemplo base o versiones extendidas de este.

Autenticación y Autorización

Muchas funcionalidades requieren que el usuario de la aplicación cliente se autentique. Se reciben el nombre de usuario y la contraseña. Normalmente, el punto de acceso de autenticación podría devolver un toquen que autoriza a esa aplicación cliente a hacer peticiones al servicio REST.El token debe ser seguro. Un atacante no debería tener forma de generar un token arbitrario válido.

Cuando el servicio recibe una petición debe comprobar que el token es válido y si lo es, comprobar que no esté caducado. Tras eso se revisa si el usuario que ha realizado la petición está autorizado.

Autenticación en HTTP

La cabecera debe ser ASCII estádar (7 bits, del 0 al 127) si hay simbolos extraños en la contraseña puede haber problemas, por ello normalmente se codifican las contraseñas en base 64.

JSON Web Token

define un formato compacto para transmitir información en JSON de forma segura entre 2 partes. Está dividido en 3 partes

- Cabecera: Es un JSON en formato base64URL que indica el tipo de token (formada por 2 elementos: TYP(tipo) y ALG)
- Cuerpo: Es un JSON en formato base64URL que contiene el cuerpo del mensaje (Formado por sub(subject), roles(rol que tiene) y exp(contenido del mensaje))
- Firma: En formato base64URL

```
{
  "typ": "JWT"
  "alg": "HS256"
}   |
    |     {
    |      "sub":"bob"
    |      "roles":["MANAGER"]
    |      "exp": 020399
    |     }  |
    |        |
    ˇ        ˇ
Cabecera.cuerpo.firma
    |   |   |      ^
    |   |   |      |
    |   |   |   Base64URL
    |   |   |      ^
    |   |   |      |
    |   |   |      |
    ---------->Algoritmo de firma
```

Se usa el formato base64URL para evitar el uso de caracteres extraños que puedan dar problemas en la comunicación.

Tipos de campos de cabecera

- Registrados: estandarizados, aunque no son obligatorios (sub para identificar username, id.... exp para indicar fecha)
- Públicos: Los puede definir cualquiera. Se suele recomendar usar el formato definido por la IANA.
- Privados: Los puede definir cualquiera, son campos que se utilizan en un contexto local

Firma

Se puede firmar con criptografía simétrica y asimétrica. Si se usa simétrica, la firma es un MAC: Message Authentication Code, mientras que si es asimétrica se la llama firma digital, donde hay una tercera firma como conocida de no repudio, que solo conoce la parte privada, de forma que el

firmante no puede negar el haber firmado algo. La firma se suele poder validar con la clave pública.

Ejemplo

En la autenticación se comprueba si usuario y contraseña son correcto, si lo son, se toman el rol, usuario y demás y se generan la cabecera y el mensaje. En caso de no ser correcto se devuelve error 401. Si todo es correcto se nos devuelve una respuesta.

OAuth

Es un sistema de autenticación. Cuando te logueas, te manda a una página de google donde iniciar sesión, pide permisos para el inicio de sesión y acto seguido redirige a la web/aplicación a la que se quiere loguear.

Endpoints del servidor de autorización

- Authorization (HTML): Formulario de autenticación
- Token (REST/JSON): permite obtener un tonken de acceso
- Introspection (REST/JSON): permite verificar tokens

OpenID Connect

SAML

Es un protocolo de autenticación y autorización. Se basa en XML. La mayor parte de los Identity Providers (IdPs) proporcionan una implementación de SAML. Contempla varios escenarios llamados perfiles. SAML va muy ligado a Single Sing On (SSO) en aplicaciones Web. SAML también suele estar relacionado con directorios de ususarios empresariales LDAP como Active Directory. SAML se suele usar generalmente de forma corporativa, no se suele usar en internet.

Como funciona

El navegador accede a la aplicación web y, dando por hecho que no se está logueado, manda al formulario de autenticación IdPs, una vez autenticado, el servidor devuelve una respuesta HTML con un formulario con un javascript con un onload (Que el usuario no ve) que hace un submit a la aplicación web, que envía los datos del formulario, con unos campos ocultos con información sobre el usuario (Que este no ve), que se podría decir que es como un ID Token en formato XML firmado. Tras eso la aplicación redirecciona a la página que se quería loguear.

Aplicaciones nativas

SAML se diseño antes de que existieran teléfonos móviles. Existen aplicaciones de terceros que permiten utilizarlo, pero con bastantes riesgos de seguridad.

Kerberos

Kerberos es un protocolo de autenticación y autorización. Esta pensado para que una aplicación en un equipo de sobremesa lo corra contra un servicio. Se realiza una petición al servicio de autenticación, Se valida el usuario y se recibe un Ticket to Get Tickets (TGT) que se guarda en el sistema operativo. Ahora, si una aplicación cliente quisiera usar este servicio, tomaría el TGT y realizaría una petición al Ticket Granting Service (TGS). Sigue los siguientes pasos:

1. Arrancas el PC
2. Te autenticas
3. EL sistema operativo pide un TGT
4. Se recibe un TGT y se guarda en el Sistema Operativo
5. Se arranca una aplicación cliente
6. Pide el TGT al sistema operativo y un token para consumir el servicio
7. Con ese ticket se establece una conexión segura para utilizar las aplicaciones

Se podría decir que es como un Single Sing On para aplicaciones de escritorio.

SPNEGO

Kerberos no se puede usar en aplicaciones web, por eso se crea SPNEGO, que permite realizar kerberos para aplicaciones web a nivel de servidor. Se suele utilizar en el esquema Negotiate de HTTP con la cabecera Autorization. Se suele usar para autenticar un navegador que corre en windows con una aplicación web usando kerberos como esquemas de autenticación para dicha aplicación web.

1. Arrancamos el PC
2. Nos autenticamos en el equipo
3. Windows Pide el TGT al servicio de autenticación y lo guarda
4. Se arranca el navegador
5. Se intenta acceder a una aplicación web
6. Se realiza una petición
7. Se recibe un error 401: Usa esquema Negotiate con authorization
8. El navegador pide al sistema operativo el TGT
9. Recibe el TGT y pide al Ticket Granting Service (TGS) un token
10. El navegador vuelve a hacer un GET con la cabecera Negotiate seguida del token
11. La aplicación web Llama al Ticket Granting Service (TGS) para validar el token
12. Si el token es válido se da acceso a la aplicación web al usuario.

Control de acceso

Impedir a otros usuarios que puedan acceder a funcionalidades como las de administrador.

Control de acceso basado en roles

Control de acceso basado en atributos

From:
https://knoppia.net/ - **Knoppia**

Permanent link:
https://knoppia.net/doku.php?id=app:tema4&rev=1734022474

Last update: **2024/12/12 16:54**