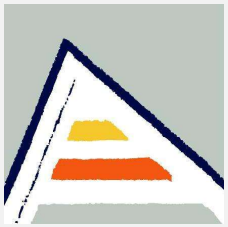


Grado en Ingeniería Informática

Sistemas distribuidos

Autenticación y autorización

Víctor Vives
vvives@dtic.ua.es

Departamento de Tecnología Informática y Computación

2021 - 2022

1. Introducción

Tanto la autenticación como la autorización tienen significados distintos

Se suele pensar que son ~~sinónimos~~

Son procesos de seguridad que tienen propósitos diferentes

Autenticación

La autenticación es el proceso por el que se verifica la identidad del usuario

Autorización

La autorización es el proceso por el que se valida a qué tiene acceso el usuario autenticado

2. Autenticación

Autenticación

La autenticación es el proceso por el que se verifica la identidad del usuario

Es el proceso por el que se identifica a un usuario y se garantiza que es quien dice ser

Evita accesos indeseados de usuarios ilegítimos al sistema

El procedimiento de autenticación más utilizado es la contraseña

La seguridad de la contraseña depende en gran medida del usuario y no del servidor

2. Autenticación

Autenticación

- **Contraseñas**: Los nombres de usuario y las contraseñas son los factores de autenticación más comunes. Si un usuario ingresa los datos correctos el sistema asume que la identidad es válida y otorga el acceso al sistema.
- **Pines de un solo uso**: Otorga acceso para una sola sesión o transacción. La autenticación sin contraseña es donde se verifica a un usuario a través de OTP o un enlace mágico entregado al correo electrónico o número de teléfono registrado.
- **Aplicaciones externas**: Genera códigos de seguridad a través de una aplicación externa que otorga el acceso al sistema. El inicio de sesión único o SSO permite a los usuarios acceder a varias aplicaciones con un único conjunto de credenciales. Autenticación mediante redes sociales.
- **Biometría**: El usuario presenta una huella digital o un escaneo facial para obtener acceso al sistema.

2. Autenticación

Contraseñas

Utilización de **sal y pimienta** para generar el hash de contraseñas.

- La sal es un valor generado aleatoriamente que se generalmente se almacena con la cadena en la base de datos diseñada para hacer imposible el uso de tablas hash para descifrar contraseñas.
- La pimienta es un valor estático de todo el sitio almacenado por separado de la base de datos (generalmente en un fichero de configuración) que está destinado a ser secreto.

El beneficio real del uso de la pimienta es evitar ataques de diccionario.

2. Autenticación

Contraseñas

Utilización de **sal** y **pimienta** para generar el hash de contraseñas.

```
# Generamos una cadena de 6 caracters
salt = random_string(6)
print salt
# '8nebce'
password = 'password123'
salted_password_hash = sha(password+salt).hexdigest()
# '26cf631a9488ed9b2135e7c43e4a4912a57f16bd'
```

```
{
  "pepper": "mysecurepepper123"
}
```

Username	Password	Salt
vvives	cbfdac6008f9cab4083784cbd1874f76618d2a97	8nebce

2. Autenticación

Pin de un solo uso o inicio sin contraseña

Envío del pin por SMS, email, aplicación de terceros, etc.

- No utilizar contraseñas implica **mayor seguridad**
- No utilizar contraseñas implica **mejor experiencia de usuario**
- No utilizar contraseñas implica **mayor productividad**
- No utilizar contraseñas implica **menos costes**

Se utiliza como factor adicional en autenticaciones multifactor (MFA).

2. Autenticación

Aplicaciones externas

- Es muy común utilizar autenticación por redes sociales.
- Los PaaS también centralizan la autenticación: Microsoft Active Directory.

Biometría

- Es muy común aprovechar el hardware de los dispositivos móviles: Touch ID o Face ID.
- Otros mecanismos más complejos también autentican mediante mano, ojos o voz. ¡DeepFake!

3. Autorización

Autorización

La autorización es el proceso por el que se valida a qué tiene acceso el usuario autenticado

Es el proceso por el que se **controla el acceso** a un usuario identificado

Evita accesos indeseados a determinados recursos del sistema

El procedimiento de autorización más utilizado es control de acceso por roles

La seguridad del control de acceso por roles la proporciona la propia compañía

3. Autorización

Autorización

- Los **controles de acceso basados en roles** (RBAC) se pueden implementar para la administración de privilegios de sistema a sistema y de usuario a sistema.
- **SAML** es un formato estándar de inicio de sesión único (SSO) en el que la información de autenticación se intercambia a través de documentos XML que están firmados digitalmente.
- El token web JSON o **JWT** es un estándar abierto para transmitir datos de forma segura entre las partes y los usuarios están autorizados mediante un par de claves pública / privada.
- La autorización **OpenID** verifica la identidad del usuario según la autenticación de un servidor de autorización.
- **OAuth** permite que la API se autentique y acceda al sistema o recurso solicitado.

3. Autorización

Control de acceso basado en roles

Los **controles de acceso basados en roles** (RBAC) se pueden implementar para la administración de privilegios de sistema a sistema y de usuario a sistema.

Se utiliza generalmente una estrategia piramidal:

1. Autorizaciones para todos los empleados
2. Pertenencia a departamentos
3. Funciones a los empleados

La creación de una estructura organizativa por roles es laboriosa: aplicación e infraestructura

3. Autorización

SAML

SAML (Security Assertion Markup Language) es un formato estándar de inicio de sesión único (SSO) en el que la información de autenticación se intercambia a través de documentos XML que están firmados digitalmente.

Transfiere los datos entre dos partes:

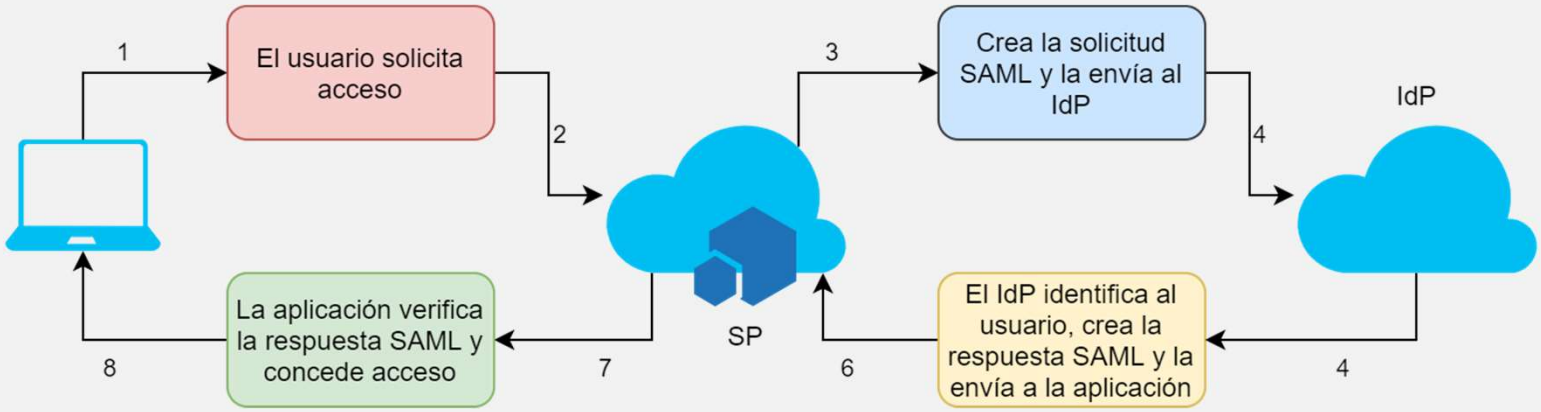
1. El **proveedor de identidades** o *identity provider* (IdP).
2. El **proveedor de servicios** o *service provider* (DP).

SAML funciona **transfiriendo la identidad** (XML firmado) del usuario de un proveedor a otro.

3. Autorización

SAML

SAML (Security Assertion Markup Language) es un formato estándar de inicio de sesión único (SSO) en el que la información de autenticación se intercambia a través de documentos XML que están firmados digitalmente.



3. Autorización

JSON web token o JWT

El token web JSON o **JWT** es un estándar abierto para transmitir datos de forma segura entre las partes y los usuarios están autorizados mediante un par de claves pública / privada.

Se componen de tres partes: **encabezado** (header), **contenido** (payload) y **firma** (signature).

Podéis probar el debugger de JWT aquí. Existen páginas para decodificar este tipo de tokens.

JWT se utiliza también para autenticar, pero el escenario más común es la autorización.

- Cada solicitud o *request* incluye el JWT, permitiendo así el acceso a rutas, servicios o recursos permitidos por ese token.

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

3. Autorización

OpenID

La autorización **OpenID** verifica la identidad del usuario según la autenticación de un servidor de autorización. El usuario se identifica a través de una URL o un XRI (eXtensible Resource Identifier).

La sintaxis de XRI 2.0 sigue el siguiente formato: `xri://host/resource:id`

OpenID es un protocolo utilizado para la **autenticación descentralizada**.

La seguridad de una conexión OpenID depende de la confianza con el proveedor de identidad.

OpenID comunica la identidad y no los permisos: ejemplo de login con Facebook o Meta.

3. Autorización

OAuth

OAuth es un estándar abierto que permite que una API se autentique y acceda al sistema o recurso solicitado. **Permite flujos simples de autorización**: aplicaciones web, de escritorio y móviles.

OpenID Connect es una **capa de identidad** sobre el **protocolo OAuth 2.0**.

Permite a los clientes verificar la identidad de un usuario basada en la autenticación realizada por un servidor de autorización así como la obtención de información del usuario utilizando REST y JSON.

¿Quieres hacer login en mi sitio web haciendo login en otro sitio web?

3. Autorización

OAuth

Las APIs más utilizadas de las grandes tecnológicas admiten OAuth 2.0:

- La Graph API de **Meta** solo admite OAuth 2.0.
- **Google** admite OAuth 2.0 como mecanismo de autorización recomendado para todas sus APIs.
- **Microsoft** admite OAuth 2.0 para varias de sus APIs y su servicio de Azure Active Directory.
- **GitHub** solo admite OAuth 2.0 como mecanismo de autorización para sus APIs.

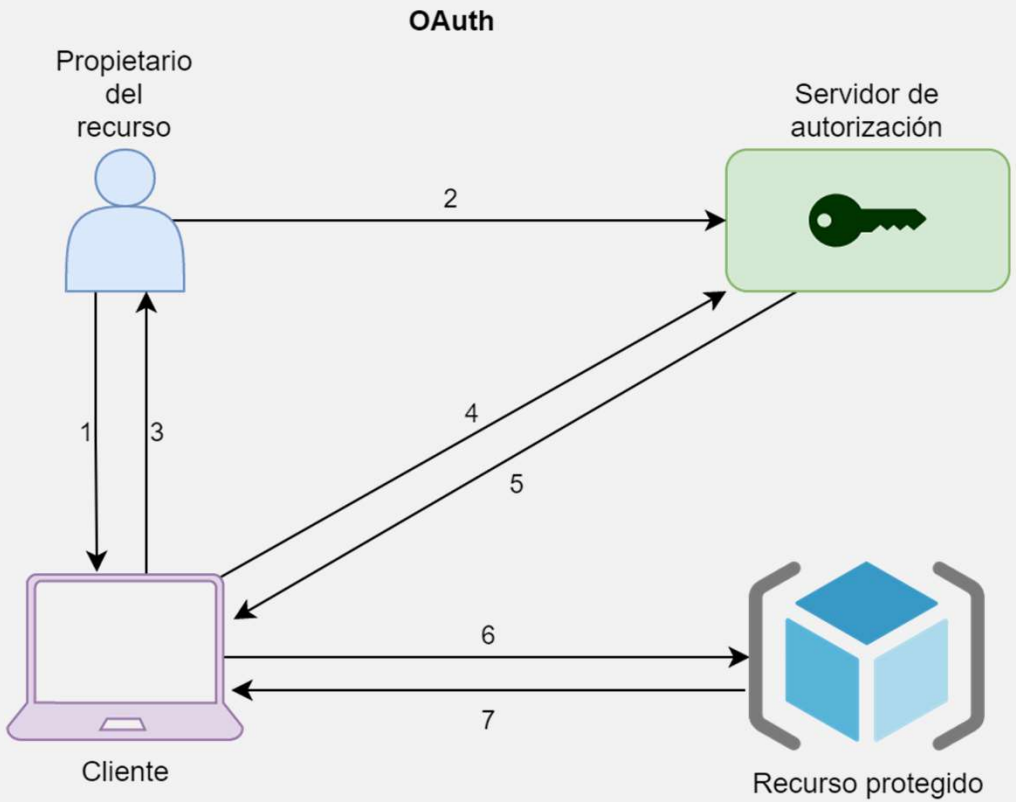
3. Autorización

OAuth

Entidades involucradas en OAuth:

1. **Recurso protegido**: Es el **recurso al que queremos acceder**.
2. **Cliente**: La aplicación que quiere acceder al recurso que está protegido, en nombre de alguien. Este cliente puede ser una aplicación web, móvil, escritorio, Smart TV, un dispositivo embebido, etc.
3. **Propietario del recurso**: Se trata del **usuario**. Se le llama el propietario de los recursos porque, si bien la API no es tuya, los datos que maneja sí lo son.
4. **Servidor de autorización**: es el responsable de gestionar las peticiones de autorización.

3. Autorización



1. Petición de autenticación
2. Identificación y consentimiento
3. Código de autorización concedido
4. Solicitud de un token con el código de autorización
5. El servidor de autorización devuelve un token
6. Petición al recurso protegido utilizando el token
7. Respuesta con los datos solicitados

3. Autorización

OAuth

Endpoints en el servidor de autorización

Para los pasos 2 y 4 disponemos de dos endpoints o URLs en el servidor de autorización.

- **Authorization** endpoint (`/authorize`): se utiliza para la interacción con los usuarios cuando éste tiene que identificarse.
- **Token** endpoint (`/token`): sólo para máquinas, sin interacción del usuario.

Ambos endpoints usan TLS y ambos deben ser conocidos por la aplicación cliente.

Los *scopes* identifican los **permisos** para el recurso protegido: lectura, escritura o ambas.

3. Autorización

OAuth

Tipos de clientes

OAuth 2.0 reconoce dos tipos de clientes.

- **Clientes confidenciales**: son aquellos que son capaces de guardar una contraseña sin que esta sea expuesta.
- **Clientes públicos**: son aquellos que no pueden mantener una contraseña a salvo.

Dependiendo del tipo de cliente valoraremos distintas formas de obtener el token de acceso.

Estas formas se llaman *flows* y se debe seleccionar una dependiendo de la aplicación que tengas.

3. Autorización

OAuth

OAuth flows

- **Authorization Code Flow**: este es el flujo más completo y seguro.
- **Implicit Flow**: se utiliza en casos en los que el usuario puede llegar a ver el *client secret* ya que no hay forma de ocultar el mismo.
- **Client Credentials Flow**: ¿Y si la aplicación no tiene usuarios? Comunicación máquina-máquina.
- **Resource Owner Password Credentials (ROPC) Flow**: Pensado para soluciones *legacy*.
- **Device Code Flow**: Extensión para casos en los que el dispositivo no tiene navegador.

3. Autorización

OAuth

Authorization Code Flow

Flujo para aplicaciones que tienen un backend seguro.

Primero se redirige al usuario al endpoint de autorización con una serie de parámetros:

```
https://authorization.server.com/authorize
?response_type=id_token
&client_id=213f6de8-f232-4854-8c20-80a9b385cca7
&redirect_uri=https://client.example.com/callback
&state=abc
&scope=returngis_api.read api2.readAndWrite
```

```
https://client.example.com/callback
?code=xxxxxxxxxx
&state=abc
```

3. Autorización

OAuth

Authorization Code Flow

Con el código obtenido la aplicación realizará una llamada POST al servidor de autorización.

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=xxxxxxxxxx
&redirect_uri=https://client.example.com/callback
&client_id=213f6de8-f232-4854-8c20-80a9b385cca7
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
   "access_token" : "una cadena muy larga",
   "token_type" : "Bearer",
   "expires_in" : 3600,
   "scope" : "returngis_api.read api2.readAndWrite"
}
```

3. Autorización

OAuth

¿Qué flujo de OAuth debería seguir?

- **Aplicaciones con backend**: Authorization Code Flow.
- **Aplicaciones sin backend**: Implicit Flow.
- **Aplicaciones nativas**: Authorization Code Flow.
- **Aplicaciones sin usuarios**: Client Credentials Flow.
- **Dispositivos sin navegador**: Device Code Flow.

En algunos casos concretos es posible utilizar flujos híbridos.

3. Autorización

OAuth

¿Cómo encajan OpenID Connect y JWT en OAuth?

OAuth solo es un framework de autorización y no es capaz de identificar usuarios.

OpenID añade las siguientes funcionalidades para complementar OAuth.

- Un **ID token** que nos permite saber quién es el usuario (JWT).
- Un nuevo endpoint **UserInfo** que nos permite recuperar más información del usuario.
- Un conjunto de *scopes* estándar y de *claims* para obtener más información del usuario.

3. Autorización

OAuth

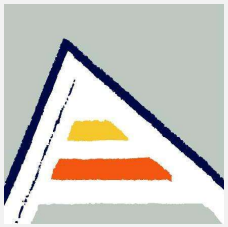
¿Cómo encajan OpenID Connect y JWT en OAuth?

Además del token de acceso obtenemos un nuevo token ID gracias a OpenID Connect.

Esta información puede ser verificada ya que normalmente está firmada digitalmente.

```
HTTP/1.1 200 OK
Content-Type: application/json
{
   "access_token" : "una cadena muy larga",
   "token_type" : "Bearer",
   "expires_in" : 3600,
   "scope" : "returngis_api.read api2.readAndWrite"
}
```

```
{
  "access_token": "fFAGRNJru1FTz70BzhT3Zg",
  "id_token": "eyJraBsdsdw3F...",
  "token_type": "Bearer",
  "expires_in": 3920,
  "scope": "returngis_api.read api2.readAndWrite"
}
```



Grado en Ingeniería Informática

Sistemas distribuidos

Autenticación y autorización

Víctor Vives
vvives@dtic.ua.es

Departamento de Tecnología Informática y Computación

2021 - 2022