Charles Olivier-Anclin
charles.olivier-anclin@uca.fr
perso.isima.fr/~chaolivi
Clermont Auvergne INP, Université Clermont Auvergne

ZZ2 F5 - Sécurité des applications web

Authentification

Plan du cours

Authentification

- Définition
- Techniques du world wide web
  - Auth Basic / Digest
  - Formulaire
  - oAuth
  - SSO, JWT
  - 2FA
- Se protéger

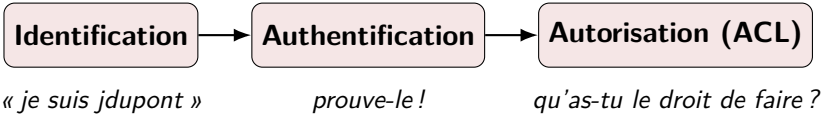
Définition

Authentification

"processus permettant à un système de s'assurer de la légitimité de la demande d'accès faite par une entité (être humain ou un autre système...) afin d'autoriser l'accès de cette entité à des ressources du système (systèmes, réseaux, applications...) conformément au paramétrage du contrôle d'accès."

Wikipedia

Trois étapes successives, souvent confondues :



- **Identification** : *se déclarer*, prétendre une identité (nom d'utilisateur, email…) — en soi, ne prouve rien
- **Authentification** : apporter la preuve de cette identité (mot de passe, token, biométrie…)
- **Autorisation / ACL** : une fois authentifié, décider quelles ressources sont accessibles (listes de contrôle d'accès, rôles…)

Preuves

Comment prouver son identité ?

- **ce que je sais** : mot de passe, numéro d'identification personnel
- **ce que je possède** : carte d'identité, carte à puce, certificat électronique, Token, Token OTP, périphérique
- **ce que je suis** : photo, caractéristique physique, voire biométrie
- **ce que je sais faire** : geste, signature

Méthode d'authentifications basiques

- Basic
- Hash
- API key (token)
- Cookies de sessions

Méthode d'authentifications basé sur les tokens

- Tokens (JWT)
- Access and refresh tokens

oAuth2 et OIDC

- oAuth2
- OpenID Connect

SSO et protocoles d'identification

- SSO (single sign-on)

Login / mot de passe

A base de login / mot de passe

La technique la plus répandue, déclinée de plusieurs façons :

- Auth **Basic** / **Digest** (niveau serveur HTTP)
- Par **formulaire** (niveau application)
- oAuth, SSO, JWT, 2FA

Apache & .htaccess

Surcharge du vhost courant via le fichier `.htaccess` (déconseillé)

```
$ sudo vi /etc/apache2/apache2.conf

<Directory /var/www/>
  Options Indexes FollowSymLinks
  AllowOverride All # None par défaut
  Require all granted
</Directory>
```

puis recharger : `sudo systemctl reload apache2`

Auth Basic / .htaccess

```
$ vi /var/www/http-basic/.htaccess

AuthType Basic
AuthName "Big Basic Secret"
AuthUserFile /var/www/html/http-basic/.htpasswd
Require valid-user
```

htpasswd, outil de création des utilisateurs

```
$ htpasswd -cB /var/www/html/http-basic/.htpasswd guest
$ htpasswd -B /var/www/html/http-basic/.htpasswd otheruser
```

mot de passe : guest

Auth Basic / requête

```
curl -i http://localhost/http-basic/
```

Le client demande un contenu (protégé)

```
GET /http-basic/ HTTP/1.1
Host: go.od
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0)
Accept: text/html,application/xhtml+xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
```

Auth Basic / réponse

Le serveur répond que le contenu est protégé

```
HTTP/1.1 401 Unauthorized
Date: Wed, 21 Oct 2015 13:26:29 GMT
Server: Apache/2.4.66 (Ubuntu)
WWW-Authenticate: Basic realm="Big Basic Secret"
Content-Type: text/html; charset=iso-8859-1
```

Auth Basic

Le client renvoie ses identifiants, encodés en base64

```
GET /http-basic/ HTTP/1.1
Host: go.od
Connection: keep-alive
Authorization: Basic b3dhc3A6cGFzc3dvcmQ=
```

- `base64(username:password)`
- Pas de chiffrement, seulement de l'encodage

Auth Digest

Même schéma que la version **Basic** excepté :

- algorithme de hashage MD5 plutôt qu'un simple encodage base64
- ajout d'un nonce, rattaché à la session de communication, pour éviter les attaques "par rejeu"

Auth Digest / activation

```
$ sudo a2enmod auth_digest
$ sudo systemctl reload apache2
```

Revenir en arrière :

```
$ sudo a2dismod auth_digest
$ sudo systemctl reload apache2
```

Auth Digest / .htaccess

```
$ vi /var/www/http-digest/.htaccess

AuthType Digest
AuthName "authdigest"
AuthDigestProvider file
AuthUserFile "/var/www/http-digest/.htdigest"
Require valid-user
```

htdigest, outil de création des utilisateurs

```
$ htdigest -c /var/www/http-digest/.htdigest authdigest guest
$ htdigest /var/www/http-digest/.htdigest authdigest otheruser
```

Auth Digest / réponse

Le serveur répond que le contenu est protégé, en indiquant le challenge

```
HTTP/1.1 401 Unauthorized
Server: Apache/2.4.66 (Ubuntu)
WWW-Authenticate: Digest realm="authdigest",
  nonce="dOA27gxfBgA=2f240f2d8a29af8f...",
  algorithm=MD5, domain="/", qop="auth"
```

Auth Digest / requête

```
GET /http-digest/ HTTP/1.1
Host: go.od
Authorization: Digest username="guest",
  realm="authdigest", nonce="6C437gxfBgA=d99239cd...",
  uri="/http-digest/", cnonce="JO8qOhvOLepFDKnQ",
  nc=00000001, qop=auth,
  response="181cdbc08b79ed2bf022c58a4bb7e64f",
  algorithm=MD5
```

Auth Digest / notes

- haché, non réversible
- SHA-256 non supporté par Apache (`htdigest`, `AuthDigestAlgorithm` limité à `MD5|MD5-sess`)
- le **nonce** évite le rejeu : le serveur connaît les nonces qu'il a émis et sait qu'ils ne sont valables qu'une fois
- le **cnonce** agit comme un *salt* : évite le cassage du mot de passe par dictionnaire
- toujours pas de verrouillage (lockout) : attaque testable à l'infini
- vulnérable au **man-in-the-middle**

Authentification par formulaire

```
<form method="POST" action="login">
  <input type="text"      name="username">
  <input type="password" name="password">
</form>

$connect = mysqli_connect("host","user","pwd", "db");
$username = $_POST['username'];
$result = $connect->query(
  "SELECT password FROM user WHERE Username = '$username';"
);
if(mysqli_num_rows($result) > 0 ){
  $row = mysqli_fetch_assoc($result);
  if(password_verify($_POST['password'], $row["password"]))
    { echo "auth success"; }
  else { echo "auth failed"; }}
else {echo "auth failed";}
```

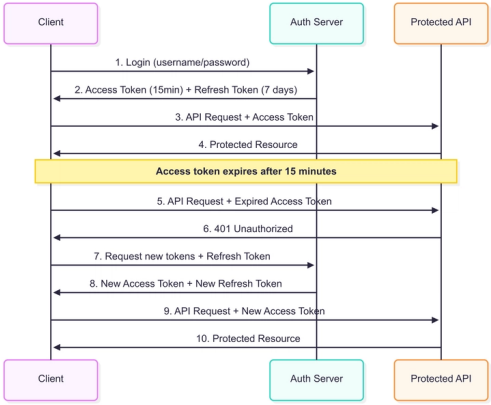
Authentification par formulaire / bonne pratique

- le mot de passe est transmis en clair au serveur
  - `HTTPS` :)
- stocker les mots de passe en clair sur le serveur est une mauvaise idée
  - Rockyou.com en 2010
- stocker les **hashés** est une bonne pratique
- **saler** les hashés est encore mieux

Section 3 : **Authentification avec des token**

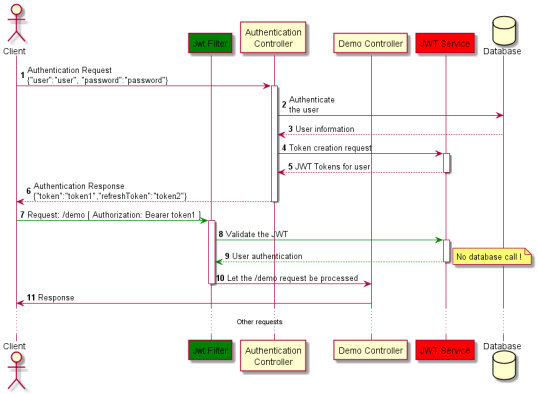
Token d'actualisation d'accès

- **1** token pour l'**accès - expiration courte**
- **1** token d'**actualisation - expiration longue**



JSON Web Token (jwt.io)

Token signé permettant de prouver une authentification, transporté dans les requêtes HTTP.



https://blog.ippon.fr/2017/10/12/preuve-dauthentification-avec-jwt/

Section 4 : **OAuth**



Version	SSL	Token	Technique
1.0	optionnel	sans expiration	mise en place compliquée
2.0	obligatoire	révocable	simple et plus léger

Standard (industriel) pour l'autorisation

Accorder à une application (client application) avec un accès limité sur une ressource (ressource server), avec le consentement du propriétaire de celle-ci (ressource owner).



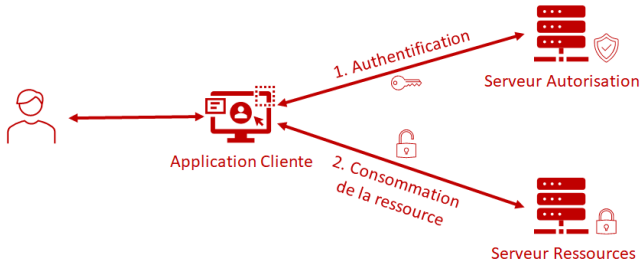
IETF Web Authorization Protocol (oauth) groupe

oAuth - Roles

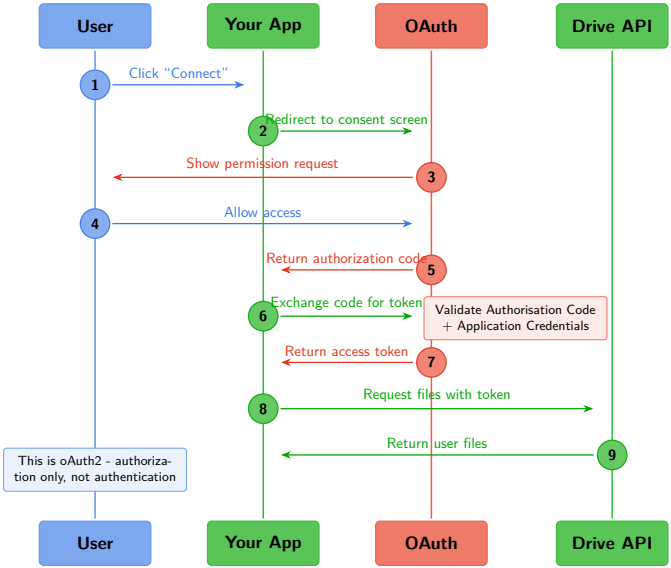
OAuth2 définit 4 rôles distincts :

- **Resource Owner :** L'utilisateur qui a les privilèges sur les données (ex : votre compte bancaire)
- **Resource Server :** Le serveur qui stocke les données (ex : le serveur de votre banque)
- **Client Application :** L'application consommatrice de la ressource (ex : application mobile)
- **Authorization Server :** Le serveur qui délivre les token d'accès à l'application.

2 phases



oAuth - flow : application Drive

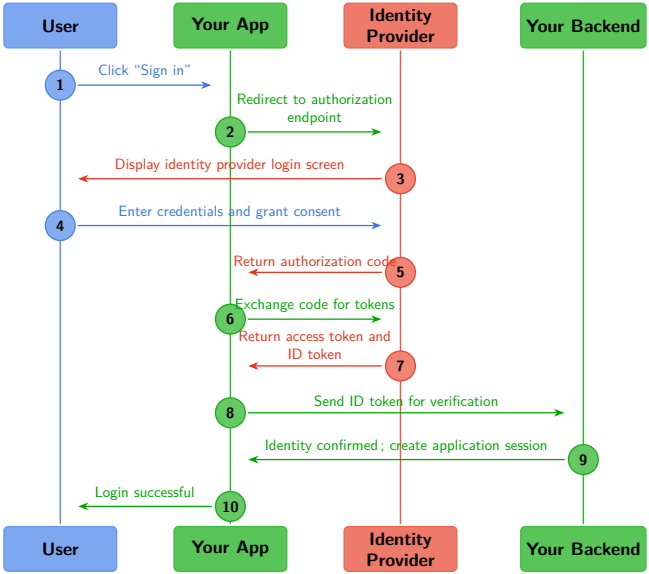


OpenID Connect (OIDC)

Couche d'identité par-dessus oAuth 2.0 : oAuth délègue l'*autorisation*, OIDC ajoute l'*authentification*.

- le client reçoit un **ID Token** (JWT) en plus de l'*access token*
- fournit l'identité de l'utilisateur : `id_token`, `userinfo` endpoint
- très utilisé pour le **SSO** : "Se connecter avec Google", Keycloak...
- openid.net — how connect works

OpenID Connect / déroulement



Single Sign-On

"Identity providers (IdPs) supply user information, while service providers (SPs) consume this information and give access to secure content"

- CAS
- Keycloak (getting started)
- OpenID
- SAML / LemonLDAP::NG

Keycloak

- Créer un Realm
- Créer un/des utilisateurs
- Modifier leurs informations
- Définir leurs mots de passe
- Tester la connection sur `localhost:8080/realms/secweb/account/`
- Déconnectez alice depuis la console d'administration
- Créer deux groupes, affecter les utilisateurs aux groupes
- Créer des roles et les affecter à des utilisateurs
- Affecter un rôle à un groupe
- Vérifier les droits hérités

Second facteur d'authentification

Confirmer l'identité par un deuxième facteur, en plus du mot de passe.



https://www.google.com/landing/2step/

WebAuthn



Yubico

Se protéger

- bien **choisir** son système d'authentification
- bien **implémenter** son système d'authentification
  - ESAPI (The OWASP Enterprise Security API)
  - libs & bundles éprouvés
- mettre en place un **deuxième facteur** d'authentification
- utiliser la **crypto asymétrique**