

# INF5190 - Authentification et identification

Jean-Philippe Caissy

2 octobre 2019

# Identification

Objectif : informer l'identité d'un utilisateur/système/client/etc

Exemples d'identifications :

- ▶ Entrer son nom d'utilisateur
- ▶ L'entête HTTP User-Agent
- ▶ Une carte de débit
- ▶ Une carte d'identité

L'identification est la première étape pour valider l'identité

# Authentication

Objectif : valider et vérifier l'identité fournie d'un utilisateur/système/client/etc

Exemples d'authentications :

- ▶ Le mot de passe d'un identifiant
- ▶ Le NIP d'une carte de débit
- ▶ La validation de la photo et de la signature d'une carte d'identité

# Autorisation

Une fois l'identité fournie, et validé par une authentification cela permet de confirmer l'autorisation à une ressource

Exemple d'autorisation :

- ▶ Seul le propriétaire d'un compte bancaire peut y avoir accès
- ▶ Un administrateur peut posséder les autorisations nécessaire pour voir les courriels d'un membre de l'organisation

# Authentification

## Types

L'authentification d'une identité se fait à partir d'un des 4 éléments suivants :

- ▶ Connaissance

Seulement un utilisateur légitime peut connaître une information secrète que le système distant sait également. Par exemple, un mot de passe, ou une question secrète.

- ▶ Possession

L'utilisateur possède un item pouvant validé son identité. Par exemple : une carte d'identité, une clé physique, une carte d'accès (*key fob*), authentifieur à code uniques

- ▶ Identitaire

Aspects de l'utilisateur qui peut difficilement être forgé. Il s'agit surtout des identifications biométriques : emprente digitale, l'iris des yeux, validation de la voie

# Authentication

## Multi-facteurs

- ▶ *single factor, two factor, multi-factor authentication*

On peut authentifier un identifiant avec un, ou plusieurs mécanismes d'authentification. Le nombre de mécanisme d'authentifications utilisés représente le nombre de facteurs utilisés.

- ▶ Facteur d'authentification unique

Valider l'identification avec un seul mécanisme d'authentification

- ▶ Facteur d'authentification double

Valider l'identification avec deux mécanismes d'authentifications

- ▶ Facteur d'authentification multiple

Valider l'identification avec plus de deux mécanismes d'authentifications

# Authentification

## Multi-facteurs

- ▶ Facteur d'authentification unique

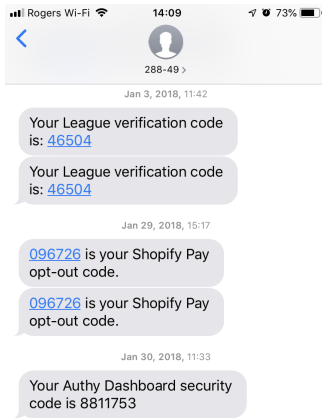
Courrier UQÀM, n'importe quel formulaire de connexion avec seulement un identifiant et mot de passe

# Authentication

## Multi-facteurs

### ► Facteur d'authentification double

Se connecter à un site Internet avec nom d'utilisateur et mot de passe (connaissance) et fournir un code unique envoyé par texto (possession).





# Authentication

## Multi-facteurs

- ▶ Facteur d'authentification multiples

Une entrée de bâtiment nécessitant une carte d'accès (possession), un NIP (connaissance) et une empreinte digitale (identitaire)

# Authentication

## Forces

Il existe deux forces d'authentifications : **faible** et **fortes**.

**N.B. : une authentification forte n'est pas nécessairement une authentification multi-facteurs**

- ▶ Faible

Une authentification faible représente un scénario où la force d'authentification est faible par rapport à l'entité que l'authentification protège.

- ▶ Forte

À l'inverse, une authentification forte utilisera un ou plusieurs mécanismes d'authentifications afin de protéger adéquatement l'entité derrière le système d'authentification.

# Authentification

## Mot de passe

- ▶ La combinaison identifiant / mot de passe est de loin le mécanisme d'authentification le plus populaire pour une application Web.
- ▶ Le mot de passe est quelque chose que l'on connaît, mais que l'application distante doit connaître également
  - ▶ Le mot de passe doit donc être stocké par l'application Web

# Authentication

## Mot de passe

### **Ne jamais stocker les mots de passes**

Jamais. Point final.

- ▶ Si l'application Web est compromise, un attaquant peut utiliser les informations d'identification et de mot de passe pour s'authentifier en tant que n'importe quel utilisateur
- ▶ Beaucoup de gens ré-utilisent leur mot de passe à travers différentes applications

# Authentication

## Mot de passe

```
';--have i been pwned?  
https://haveibeenpwned.com/
```

# Authentication

## Mot de passe

## Hachage

Fonction de hachage : à partir d'une donnée fournie en entrée, retourner une empreinte numérique afin d'identifier rapidement la donnée initiale.

Les fonctions de hachages sont utilisés pour reconnaître rapidement des fichiers ou des mots de passes.

Exemple :

```
hash("Renard") => "62b61bc0"
```

```
hash("renard") => "ad0b225c"
```

```
hash("Le renard") => "b8e9d4c7"
```

# Authentication

Mot de passe

Hachage

- ▶ Fonction à sens unique
- ▶ Le calcul doit être rapide et facile
- ▶ L'inverse de la fonction de hachage est infaisable par calcul

Exemples de fonctions de hachage :

- ▶ md5
- ▶ sha-1
- ▶ whirlpool
- ▶ blowfish

# Authentication

Mot de passe

Hachage

Contraintes d'une fonction de hachage :

- ▶ Collisions : deux entrées de données différents donnant le même résultat avec la fonction de hachage

Exemple de collision : <https://shattered.it/>

```
$ md5sum extra/*
```

```
5bd9d8cab46041579a311230539b8d1  extra/sah1-doc2.pdf
```

```
ee4aa52b139d925f8d8884402b0a750c  extra/sha1-doc1.pdf
```

```
$ sha1sum extra/*
```

```
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  extra/sah1-doc2.p
```

```
38762cf7f55934b34d179ae6a4c80cadccbb7f0a  extra/sha1-doc1.p
```



# Authentication

## Mot de passe

## Hachage

L'objectif d'utiliser une fonction de hachage sur un mot de passe est de limiter les dégâts en cas de fuite d'une base de donnée

- ▶ Un utilisateur malveillant peut pré-construire des tables de hachage à partir de dictionnaires, ou de mots de passes déjà disponible
- ▶ Cela permet de rapidement connaître les mots de passes qui ont déjà été utilisé

Attaque par dictionnaire : bâtir tous les `hash` possible à partir d'un dictionnaire de mots

# Authentication

## Mot de passe

## Hachage

Le salage est une méthode qui permet de renforcer la sécurité des informations hachées.

- ▶ Empêche que deux informations identiques produisent la même empreinte
- ▶ Mitige les risques pour les attaques par dictionnaire si le hash n'est pas connu de l'attaquant et chance pour chaque enregistrement

# Authentication

Mot de passe

Hachage

`hash(mot de passe + sel)`

- ▶ Le sel représente une information supplémentaire statique et générée aléatoire
- ▶ Bonne pratique : un seul sel par enregistrement (ne pas réutiliser)
  - ▶ Le sel peut être stocké en clair
  - ▶ Il est utilisé à chaque vérification d'authentification qui hash le mot de passe avec le sel

# Authentication

Mot de passe

Hachage

Table 1: Exemple du contenu d'une table utilisateur d'une BD

user	salt	hash_password
toto	KSIZIjMeH4AhTG	84d9c4cdcf4c808c9c35...
tata	envNCgmZPHOXWQ	004eaff15cfba8c7922a2...

`hash("password") => 5f4dcc3b5aa765d61d8327deb882cf99`

`hash("passwordKSIZIjMeH4AhTG") => 84d9c4cdcf4c808c9c35...`

# Authentication

## Mot de passe

## Hachage

Deuxième contrainte : complexité et temps nécessaire pour la méthode de hachage

- ▶ Pour un fichier, on veut une méthode de hachage rapide.
- ▶ Pour un mot de passe : on veut une méthode de hachage qui prends beaucoup de ressource
  - ▶ Plus difficile pour un attaquant : nécessite plus de ressources (CPU, RAM, temps)

# Authentication

## Mot de passe

## Hachage

Il existe des méthodes de hachage spécifique aux mots de passe. Ils supportent :

- ▶ L'ajout d'un sel
- ▶ Notion de complexité

La complexité est un chiffre. Plus il est élevé, plus la méthode de hachage va avoir besoin de ressources, donc de temps pour produire le hash final.

Il s'agit souvent du nombre d'itération sur la fonction de hachage.

# Authentication

Mot de passe

Hachage

Fonctions de hachage spécifique à des mots de passe :

- ▶ bcrypt
- ▶ PBKDF2

**Ces méthodes sont la norme et devraient TOUJOURS être utilisés pour des mots de passes**

# Authentication

## Mot de passe

## Usage unique

Un mot de passe à usage unique (*OTP : One Time Password*) est un mot de passe valide que pour une seule authentification.

Permet d'empêcher les attaques par rejeu commun aux mots de passes statique :

- ▶ Si un mot de passe est connu, celui-ci peut être utilisé n'importe quand par n'importe qui
- ▶ Un mot de passe unique n'est valide qu'une seule fois, et souvent pour un temps limité

Utilisations majeurs :

- ▶ Authentification d'un API (OAuth, JWT)
- ▶ Facteur d'authentification doubles avec des codes générés et valide pour un temps spécifique



# Authentication

## Mot de passe

## Usage unique

### **TOTP** : Time-based One Time Password

Mot de passe unique généré en fonction du temps actuel.

Le serveur et le client s'échangent une information secrète. Une méthode de hachage est utilisé avec comme entrée la clé secrète et l'heure actuel pour généré un code que chaque partie peut valider.

```
hash(secret + 2019/02/01 20:05:11) => 544871
```

```
hash(secret + 2019/02/01 20:05:25) => 544871
```

```
hash(secret + 2019/02/01 20:05:31) => 150986
```

```
hash(secret + 2019/02/01 20:05:55) => 150986
```

# Authentication

Mot de passe

Usage unique

Puisque les deux parties (serveur et client) connaissent et se sont échangé le secret partagé, si le temps de chacun est synchronisé un code unique peut être généré et validé par l'autre partie

# Authentication

## Application Web

Il existe plusieurs mécanismes pour un utilisateur de s'authentifier sur une application web :

- ▶ Cookie
- ▶ HTTP Header

# Authentication

## Application Web

### Cookie

- ▶ Après authentification, un cookie de session est généré
  - ▶ Permet de lier les visites subséquentes à l'utilisateur déjà authentifié
- ▶ Cookie peut posséder une date d'expiration, ou expirer à la fermeture de navigateur

# Authentication

## Application Web

## HTTP Header

Le protocole HTTP contient des mesures d'authentification de base avec l'entête `Authorization`

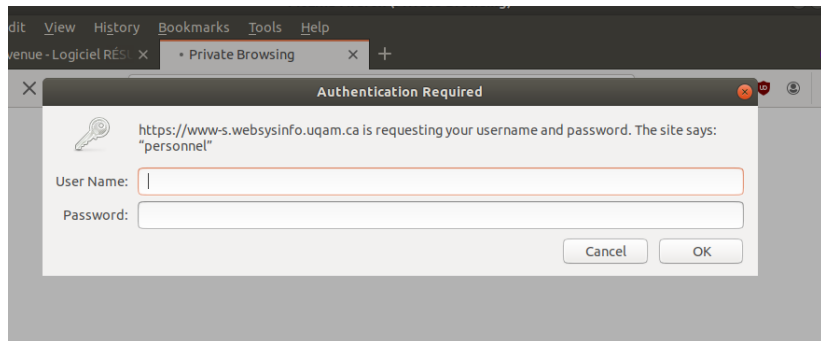


Figure 2

# Authentication

## Application Web

### HTTP Header

Le client (navigateur) doit retourner un entête nommé `Authorization` qui contient les informations suivantes :

```
Basic base64("nom d'utilisateur" + ":" + "mot de passe")
```

*Base64* : codage d'information binaire utilisant 64 caractères.

Exemple :

Texte	Encodé base64
INF5190	SU5G
INF5190	SU5GNTE5MA==
INF5190-030	SU5GNTE5MC0wMzA=

# Authentication

## Application Web

### HTTP Header

Avec le nom d'utilisateur **Aladin** et le mot de passe **Sésame**,  
**ouvre-toi** :

```
base64("Alading:Sésame, ouvre-toi") =>  
QWxhZGluZzpTw6lzYW11LCBvdXZyZS10b2k=
```

Et dans l'entête HTTP :

```
GET /protected HTTP/1.1
```

```
Host: example.com
```

```
Authorization: Basic QWxhZGluZzpTw6lzYW11LCBvdXZyZS10b2k=  
[...]
```

# Authentication

## Validation de message

Lorsqu'un message est transmis, il existe deux validations pour s'assurer que le message est authentique :

- ▶ Intégrité d'un message : s'assurer que le message n'a pas été changé
- ▶ Authenticité : s'assurer qu'un message provient de la bonne personne



# Authentication

## Validation de message

- ▶ Un usager se connecte à un service distant
- ▶ L'application web renvoie un cookie avec les données suivantes :

Set-Cookie: is\_admin=true; user=caissy.jean-philippe

N'importe qui peut rajouter is\_admin=true au cookie pour devenir administrateur!

# Authentication

## Validation de message

### MAC

Message Authentication code (MAC) : une information supplémentaire confirmant l'intégrité et l'authenticité d'un message.

- ▶ Fonctionne avec le principe d'une clé secrète que seul l'expéditeur connaît.
- ▶ La clé secrète est utilisée pour valider l'authenticité et l'intégrité d'un message
- ▶ Puisque le receveur ne connaît pas la clé secrète, il ne peut pas régénéré un MAC valide

# Authentication

## Validation de message

### MAC

Exemple :

```
secret_key = "ma clé secrète"  
cookie = "is_admin=true; user=caissy.jean-philippe;"  
mac = hash(cookie + secret_key)
```

Le MAC va être transmis avec le cookie au client

Lorsque le serveur reçoit le cookie, il va pouvoir valider que le hash du MAC correspond au cookie. Si le cookie a changé, le MAC ne correspondra pas et l'application peut ignorer le message.

# Authentication

## Application Web

### JWT

JWT : JSON Web Token

- ▶ Standard qui définit un moyen compact d'encapsuler et transmettre de manière sécuritaire des données entre deux parties en tant qu'objet JSON
- ▶ Les données sont signées (MAC)
  - ▶ Donc impossible de fausser
- ▶ Peut être chiffré
- ▶ Transmis avec l'entête Authorization

# Authentication

## Application Web

### JWT

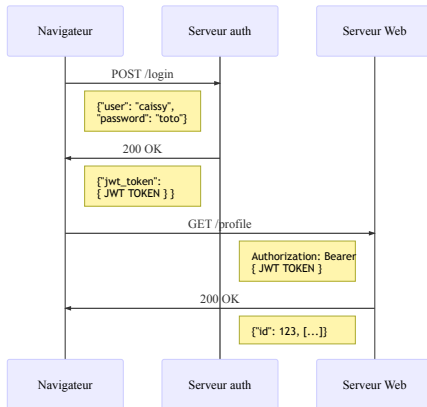
Les données contenus dans un JWT peuvent être n'importe quoi  
e.g. :

- ▶ Peut représenter l'état d'une session
  - ▶ Nom d'utilisateur
  - ▶ Permissions

# Authentication

## Application Web

### JWT



# Liens

- ▶ Introduction to JWT
- ▶ How and when to use HMAC
- ▶ What is weak authentication?