MFA

Multi Facteur Authentication

Avril 2025

Objectifs du cours

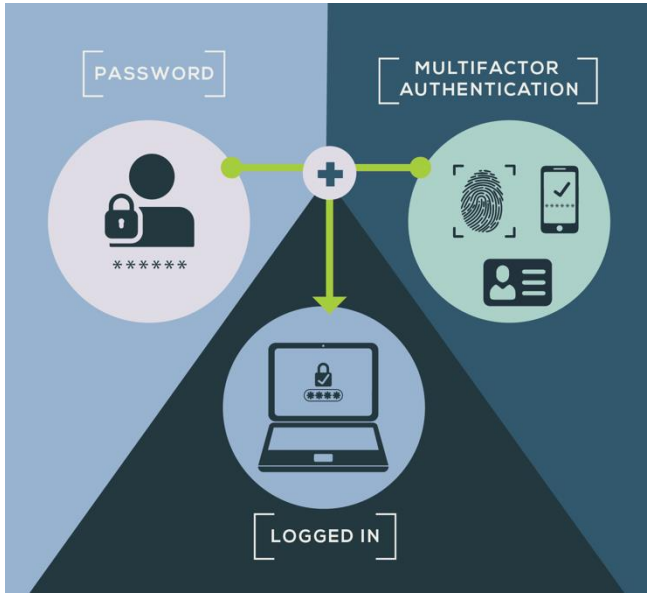
- Comprendre les enjeux de la sécurité et du MFA
- Découvrir les différentes méthodes de MFA
- Implémenter une authentification à deux facteurs dans une app web
- Savoir tester l’implementation du MFA

Introduction au MFA

Qu'est ce que le MFA

MFA = Authentification Multifacteur

Mécanisme de sécurité qui exige que l'utilisateur fournisse **au moins deux preuves d'identité différentes** avant d'accéder à son compte.



Ces preuves sont classées en trois catégories :

1. Quelque chose que je connais => un mot de passe, un code PIN
2. Quelque chose que je possède => un téléphone, une application d'authentification, une clé USB de sécurité
3. Quelque chose que je suis => une empreinte digitale, une reconnaissance faciale

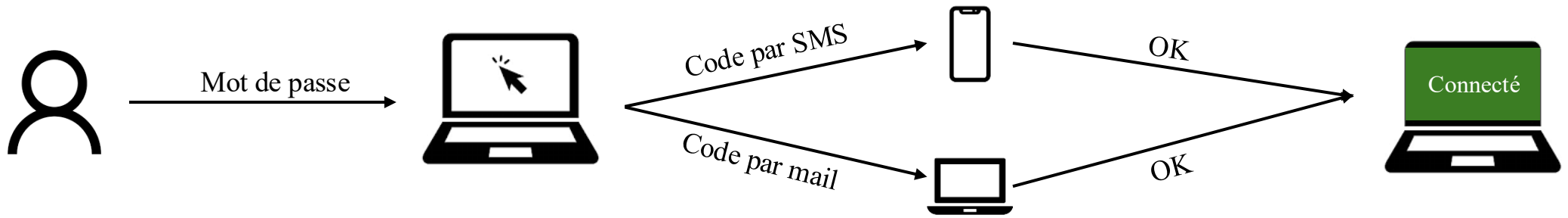
Pourquoi utiliser le MFA :

- Protéger les données personnelles et bancaires des utilisateurs
- Réduire les risques de piratage et de fraude
- Renforcer la confiance des clients
- Respecter les bonnes pratiques et réglementations en matière de cybersécurité

Les Types de MFA

1. OTP par SMS ou Email (One-Time Password)

Principe : l'utilisateur reçoit un code temporaire par SMS ou par email.



Avantages :

- Facile à mettre en œuvre
- Pas besoin d'installer une application

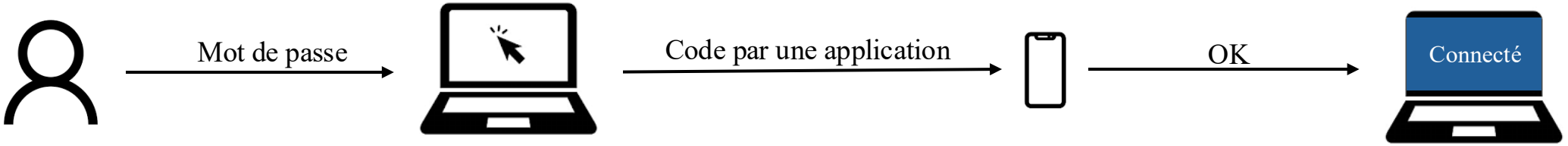
Inconvénients :

- Moins sécurisé (risque d'interception, usurpation de numéro)
- Dépendant de la connectivité réseau

Les Types de MFA

2. Time-based OTP via une application mobile (ex : Google Authenticator, Microsoft Authenticator)

Principe : une application génère un code à usage unique (One-Time Password) qui change toutes les 30 secondes.



Avantages :

- Plus sécurisé que l'OTP par SMS
- Fonctionne hors ligne

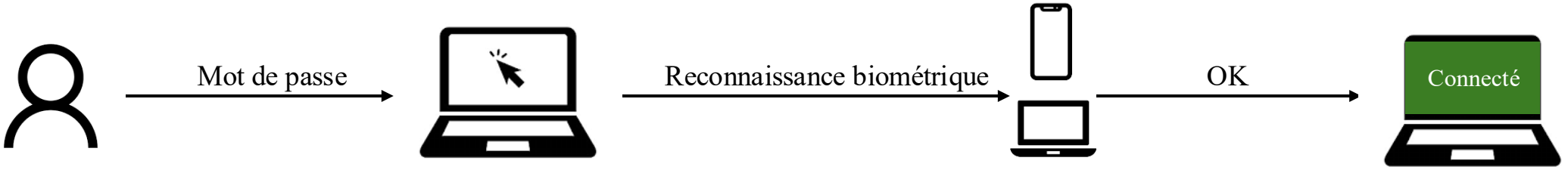
Inconvénients :

- L'utilisateur doit installer une application (Google Authenticator, Microsoft authenticator, …)
- Si le téléphone est perdu, il faut prévoir un système de récupération

Les Types de MFA

3. Authentification biométrique

Principe : reconnaissance faciale, empreinte digitale, ...



Avantages :

- Très pratique et rapide
- Haute sécurité

Inconvénients :

- Dépendant du matériel de l'utilisateur
- Complexe à implémenter côté web sans passerelle (souvent géré côté OS ou appareil mobile)

Les Types de MFA

4.Clé physique (clé de sécurité FIDO2 / YubiKey)

Principe : l'utilisateur branche une clé USB sécurisée pour valider son identité.

Avantages :

- Très haut niveau de sécurité
- Supporté par les navigateurs modernes

Inconvénients :

- Nécessite un équipement spécifique
- Moins adapté à un grand public sans formation

5.Questions de sécurité

Principe : l'utilisateur répond à une ou plusieurs questions définies lors de l'inscription (ex : "Quel est le nom de votre premier animal ?").

Avantages :

- Simple à mettre en œuvre

Inconvénients :

- Faible sécurité (souvent des réponses devinables)
- Peu recommandé dans un contexte professionnel

TP Guidé : Implémenter le MFA dans une application Node.js

Objectif :

- Comprendre le fonctionnement du MFA avec des TOTP (Time-based One-Time Passwords)
- Générer un secret et un QR Code à scanner dans une app d'authentification (Google Authenticator, Authy…)
- Vérifier le code saisi par l'utilisateur pour autoriser l'accès

Prérequis :

- Node.js installé sur votre machine
- Postman (ou équivalent) pour tester les requêtes
- Une app d'authentification installée sur votre téléphone (ex : Google Authenticator)

TP Guidé : Implémenter le MFA dans une application Node.js

1. **Lancer l'application**

```
npm init -y
npm install express speakeasy qrcode
```

2. **Génération de QRCode : Revue index.js**

3. **Etapes de tests**
   i. Lancer le serveur

```
node index.js
```

   ii. Ouvrir dans le navigateur
   http://localhost:3000/generate-mfa

   iii. Scanner le QR Code avec l'app d'authentification

TP Guidé : Implémenter le MFA dans une application Node.js

2. **Etapes de tests**
   iv. Tester le code via postman

Paramètre	Valeur
Méthode	POST
URL	http://localhost:3000/verify-mfa
Header	Content-Type: application/json
Body	{"token":"votre_code"} //remplacer par le code affiché sur votre application auth

Si le code est bon : MFA successful
Sinon : Invalid MFA code

Application : Mise en place d'un MFA TOTP

A l'inscription et à la connexion

Prérequis :

- Avoir une application mobile type **Microsoft Authenticator**
- Avoir un serveur (celui de votre application web), y ajouter la génération du code MFA (exemple nodeJS)
- Avoir une entrée dans la base de données des utilisateurs pour stoquer le secret MFA

Processus

1. À l'inscription (signup) :
    - Le serveur génère un secret MFA unique pour le nouvel utilisateur (speakeasy.generateSecret()).
    - Il sauvegarde ce secret MFA dans la base de données associé au compte de l'utilisateur.
2. À la connexion (login) :
    - Le serveur vérifie d'abord email + mot de passe.
    - Si c'est correct ➔ Il demande à l'utilisateur de saisir son code MFA à 6 chiffres.
    - Le serveur récupère le secret MFA depuis la base de données pour cet utilisateur.
    - Le serveur utilise ce secret pour calculer ce que devrait être le bon code (speakeasy.totp.verify()).
    - Il compare le code calculé et le code saisi par l'utilisateur.
    - Si ça match ➔ L'utilisateur est connecté
    - Sinon ➔ Refus de la connexion

Application : Mise en place d'un MFA TOTP

A l'inscription et à la connexion

Prérequis :

- Avoir une application mobile type **Microsoft Authenticator**
- Avoir un serveur (celui de votre application web), y ajouter la génération du code MFA (exemple nodeJS)
- Avoir une entrée dans la base de données des utilisateurs pour stoquer le secret MFA

Action	**Connaissance nécessaire**
Générer un secret MFA	Utiliser **speakeasy**
Générer un QR Code	Utiliser **qrcode**
Stocker un secret MFA	Sauvegarder dans la base de données le cas échéant
Vérifier un code MFA	speakeasy.totp.verify() dans leur backend
Ajouter un champ MFA dans le processus de login	Modifier leur page de login pour demander le code

Application : Mise en place d'un MFA TOTP

A l'inscription et à la connexion

Exemple de modèle dans une base de données

```sql
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash TEXT NOT NULL,
    mfa_enabled BOOLEAN DEFAULT FALSE,       -- Est-ce que le MFA est activé pour ce compte ?
    mfa_secret TEXT,                         -- Stockage du secret MFA (lié à l'app Authenticator)
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

NB : Penser à chiffrer le secret dans la base de données