UNIVERSITÉ
SAVOIE
MONT BLANC

Authentification à deux facteurs

Binôme

Sabrina Khelifi

Maxime Vigny

Table des matières

Table des illustrations

1. Introduction

Aujourd'hui les systèmes de sécurité sont encore imparfaits. Les mots de passe, plus longs et compliqués, créés par des générateurs comme le porte-clés iCloud de Safari ou des applications tierces comme LastPass ou 1Password, peuvent aider, mais la meilleure façon de verrouiller les accès aux données est d'ajouter des options de sécurité supplémentaires pour une vérification en plusieurs étapes afin de rendre le système plus fiable.

L'authentification est une procédure, par laquelle un système informatique certifie l'identité d'une personne ou d'un ordinateur. Le but de cette procédure étant **d'autoriser** la personne à accéder à certaines ressources sécurisées

Il existe 4 facteurs d'authentifications qui peuvent être utilisés dans le processus d'autorisation d'accès à des ressources bloquées et sécurisées :

- **Ce que l'on connaît (facteur mémoriel)** : une information que l'utilisateur a mémorisée et que lui seul connait (exemple : un mot de passe, un nom)
- **Ce que l'on possède (facteur matériel)** : une information que seule l'utilisateur possède et enregistrée dans un support (exemple : une clé USB).
- **Ce que l'on est (facteur corporel)** : une information qui caractérise l'utilisateur avec une empreinte qui lui est propre (exemple : voix, pupille, empreinte digitale)
- **Ce que l'on sait faire (facteur réactionnel) :** une information ou un geste que seul l'utilisateur peut produire (exemple : une signature)

L'authentification à 2 facteurs consiste à rajouter une étape d'authentification afin de certifier l'identité de la personne qui se connecte. On peut tout aussi bien rajouter plus d'un facteur afin d'avoir une sécurité encore plus robuste. On parle d'authentification à multiple facteurs ou MFA en anglais (Multiple FactorsAuthentication)

Il existe trois familles d'authentification : simple, forte et unique. L'**authentification simple** ne repose que sur un seul facteur alors que l'**authentification unique** permet une seule authentification permettant d'accéder à plusieurs applications informatiques. Quant à l'**authentification forte,** elle repose sur deux facteurs ou plus [1]

A travers notre travail nous allons aborder un type d'authentification forte qui est l'authentification à deux facteurs.

Par la suite, nous emploierons l'acronyme 2FA (2 factor authentication) afin de désigner l'authentification à deux facteurs.

2. Fonctionnement

2.1) Premier facteur : authentification simple

L'utilisateur va s'authentifier en rentrant son username / password auprès du service auquel il souhaite accéder. Le service va faire passer le password dans une fonction de hachage(signature) et l'envoyer au serveur du service, qui va le comparer avec celui qui a été enregistré lors de la création du compte, et, s'ils correspondent, l'utilisateur accèdera au service.

Le serveur ne va pas enregistrer le mot de passe mais enregistrera cette signature. Lorsque l'utilisateur se connectera, le serveur ne va pas vérifier si le mot de passe est identique, mais il va vérifier que la signature du mot de passe saisi est bien la même que la signature du mot de passe enregistré. Voyons ce que cela donne avec le même schéma qu'au-dessous mais avec utilisation d'une fonction de hachage

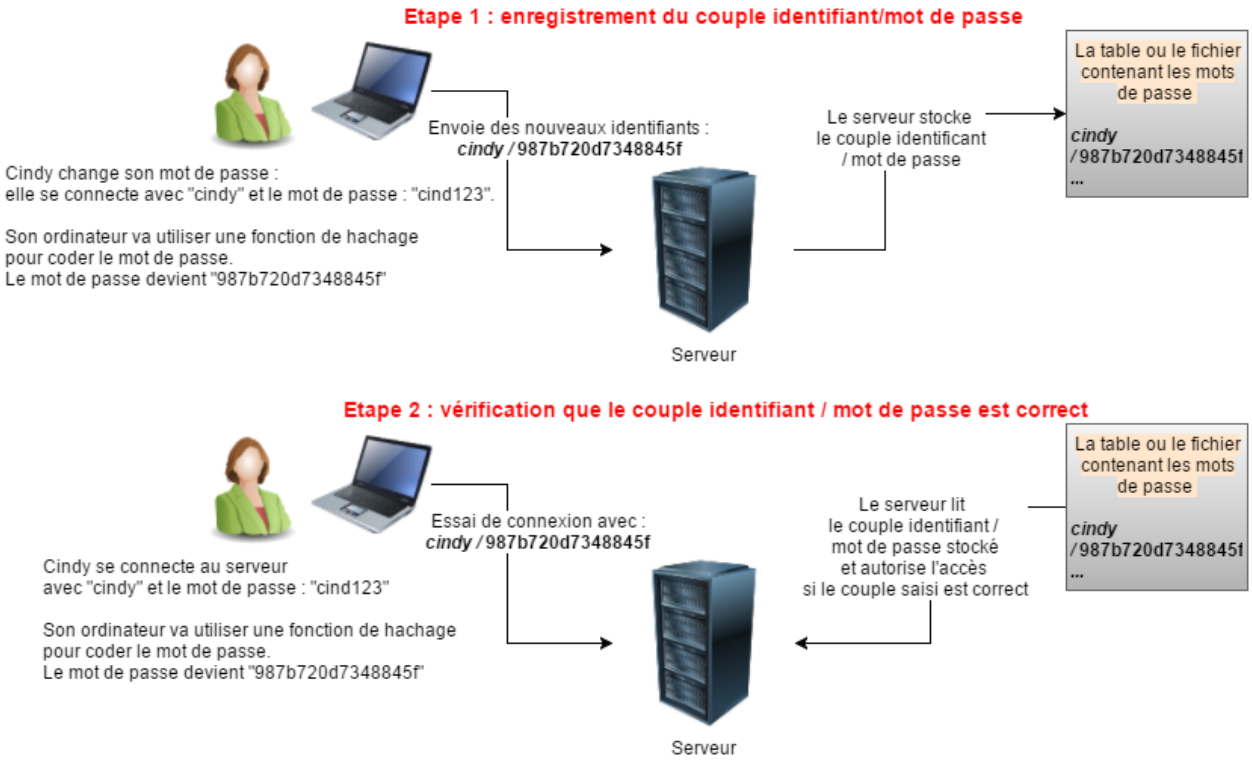

Figure 1 : Authentification simple

2.2) Deuxième facteur :

Première version :

Le schéma suivant décrit une authentification à deux facteurs qui est souvent faite avec un premier facteur qui est une authentification simple et un deuxième facteur qui est souvent matériel.

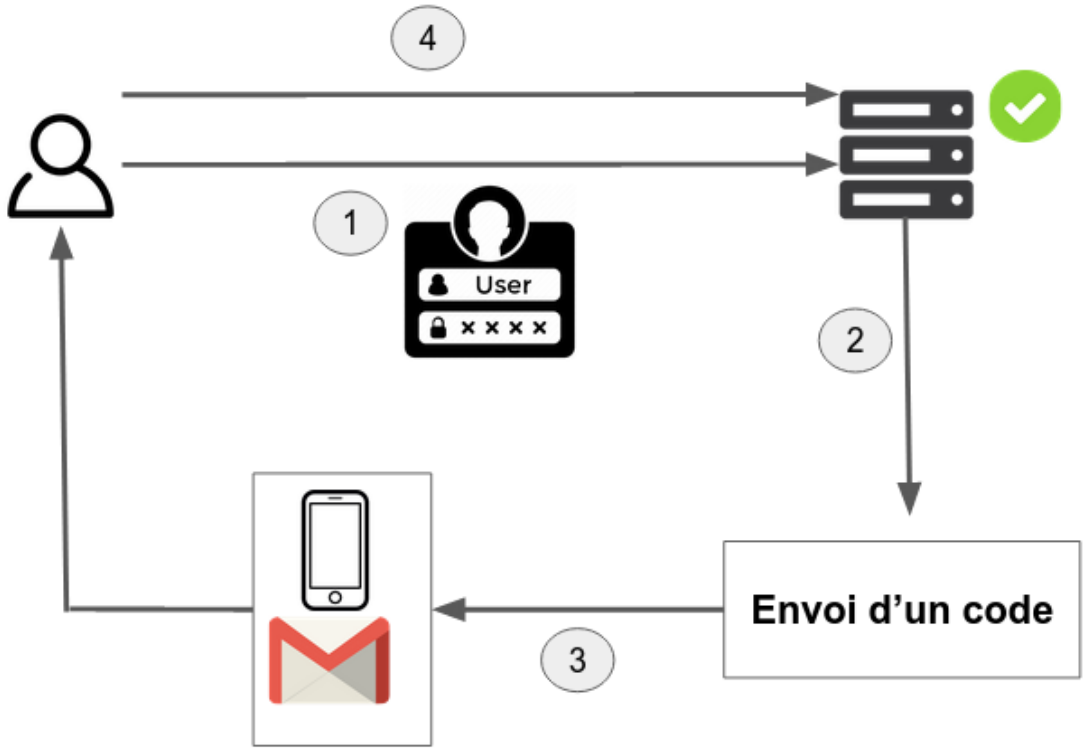

Figure 2 : Résultat exécution HOTP

1. L'utilisateur souhaite accéder à un service, auquel il se connecte avec son username / password.
2. Une fois avoir validé la première authentification de l'utilisateur, le serveur va alors créer un code, une suite de chiffres et de lettres généré aléatoirement, qui va être valide pendant une durée limitée.
3. Le serveur va ensuite envoyer à l'utilisateur le code généré via un moyen choisi au préalable : par mail, par SMS, ou avec une notification provenant d'une application mobile.
4. Ainsi, l'utilisateur va entrer le code, qui une fois validé par le serveur lui permettra de se connecter au service.

Deuxième version:

Algorithme Time-based One-time Password(TOTP)

L'algorithme TOTP implique qu'un OTP(One-time Password) est le produit de deux paramètres cryptés ensemble. Il s'agit d'une valeur commune, qui est une clé secrète partagée, et d'une variable, dans ce cas : le temps de fonctionnement. Ces paramètres sont chiffrés à l'aide d'une fonction de hachage.

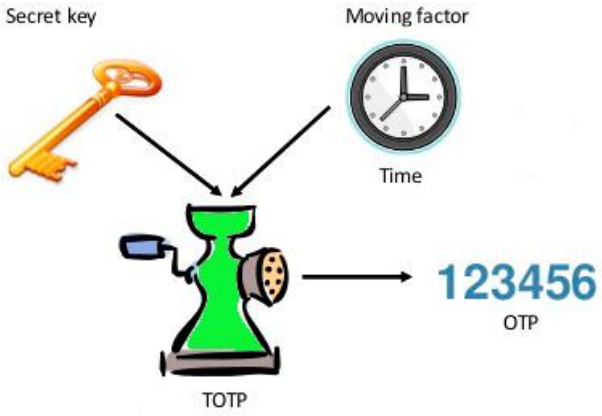

Figure 3 : Fonctionnement du TOTP

Le schéma ci-dessous illustre les étapes suivantes :

1. Un utilisateur souhaite se connecter à une application ou un site web protégé par TOTP 2FA. Pour que l'authentification OTP fonctionne, l'utilisateur et le serveur TOTP doivent initialement partager un paramètre statique (une clé secrète).

2. Lorsque le client se connecte au site web protégé, il doit confirmer qu'il possède la clé secrète. Ainsi, son jeton TOTP fusionne la clé secrète et l'intervalle de temps actuel, et génère une valeur HASH en exécutant une fonction HASH prédéterminée. Cette valeur est essentiellement le code OTP que l'utilisateur voit sur le jeton.

3. Tant que la clé secrète, la fonction HASH et le pas de temps sont les mêmes pour les deux parties. Le serveur effectue le même calcul que le générateur OTP de l'utilisateur.

4. L'utilisateur saisit l'OTP et si celui-ci est identique à la valeur du serveur, l'accès est accordé. Si les résultats des calculs ne sont pas identiques, l'accès est, bien entendu, refusé.

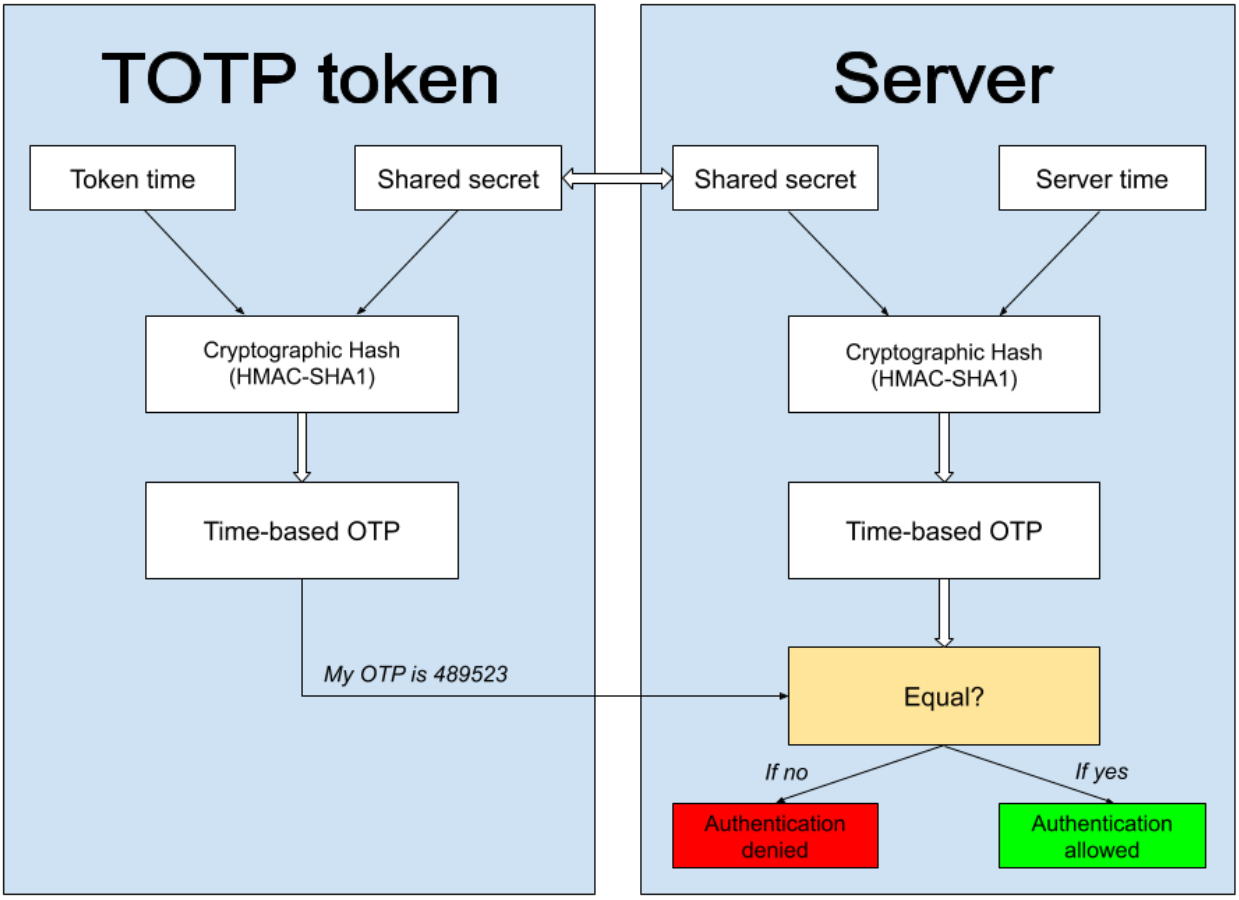

Figure 4 : Architecture TOTP

L'OTP basé sur les événements (également appelé HOT, signifiant Mot de passe à usage unique basé HMAC) est l'algorithme original utilisé pour le mot de passe à usage unique et créé à partir de deux informations. La première est la clé secrète, qui est connue uniquement du jeton(ou application générant le token) et du serveur qui valide les codes OTP soumis. La seconde information est le facteur de changement qui, pour un OTP basé sur les événements, est un compteur. Ce compteur est stocké dans le jeton et sur le serveur. Le compteur du jeton s'incrémente lorsqu'on appuie sur le bouton du jeton, tandis que le compteur du serveur s'incrémente uniquement lorsqu'un OTP est correctement validé.

Comparaison HOTP et TOTP:

HOTP génère plusieurs codes "OTP suivant" valides parce que le bouton du jeton peut être activé plusieurs fois, ce qui incrémente le compteur du jeton sans que l'OTP créé soit envoyé au serveur de validation. Pour cette raison, les serveurs de validation HOTP acceptent une série d'OTP. Pour être plus précis, ils accepteront un OTP généré par un compteur et qui se situe dans une plage déterminée d'incréments après la valeur de compteur précédente stockée sur le serveur. Cette plage s'appelle la fenêtre de validation. Si le compteur du jeton se trouve hors de la plage autorisée par le serveur, la validation échoue et le jeton doit être resynchronisé.

Ainsi, avec HOTP, il faut clairement faire un compromis. Plus la fenêtre de validation est large, moins il y aura de chance de devoir resynchroniser le jeton avec le serveur, ce qui n'est pas pratique pour l'utilisateur. Mais l'élément le plus important est que plus la fenêtre est large, plus il y aura de chance qu'une attaque par force brute devine l'un des OTP acceptés.

Cependant, avec TOTP, il n'existe qu'un seul OTP valide à un quelconque moment : celui généré par l'heure UNIX courante.

Formalisme de l'algorithme :

$$X = HMAC(K, CT)$$

X est le résultat d'un HMAC de SHA1 calculé à partir d'une clé secrète K et un compteur CT.

$$Sbits = tronquer(X)$$

Tronquer va extraire 31 bits à partir de X.

$$T = dec(Sbits)\ _{mod 10^D}$$

Onprend ces bits convertis en décimal auquel on applique le modulo 10^D .

Ceci nous donne notre token. D étant la longueur de la chaîne.

Le Compteur CT se calcule comme suit :

$$C_T = \left\lfloor \frac{T - T_0}{T_X} \right\rfloor$$

T est le temps actuel en secondes depuis une époque particulière.

T0 est l'époque spécifiée en secondes depuis l'époque Unix (par exemple, si l'on utilise le temps Unix, alors T0 est 0).

TX est la durée d'un temps (par exemple 30 secondes).

Activation et enregistrement avec TOTP 2FA :

Nous allons prendre l'exemple suivant :

Alice n'a besoind'installer l'application d'authentification sur son téléphone qu'une seule fois.

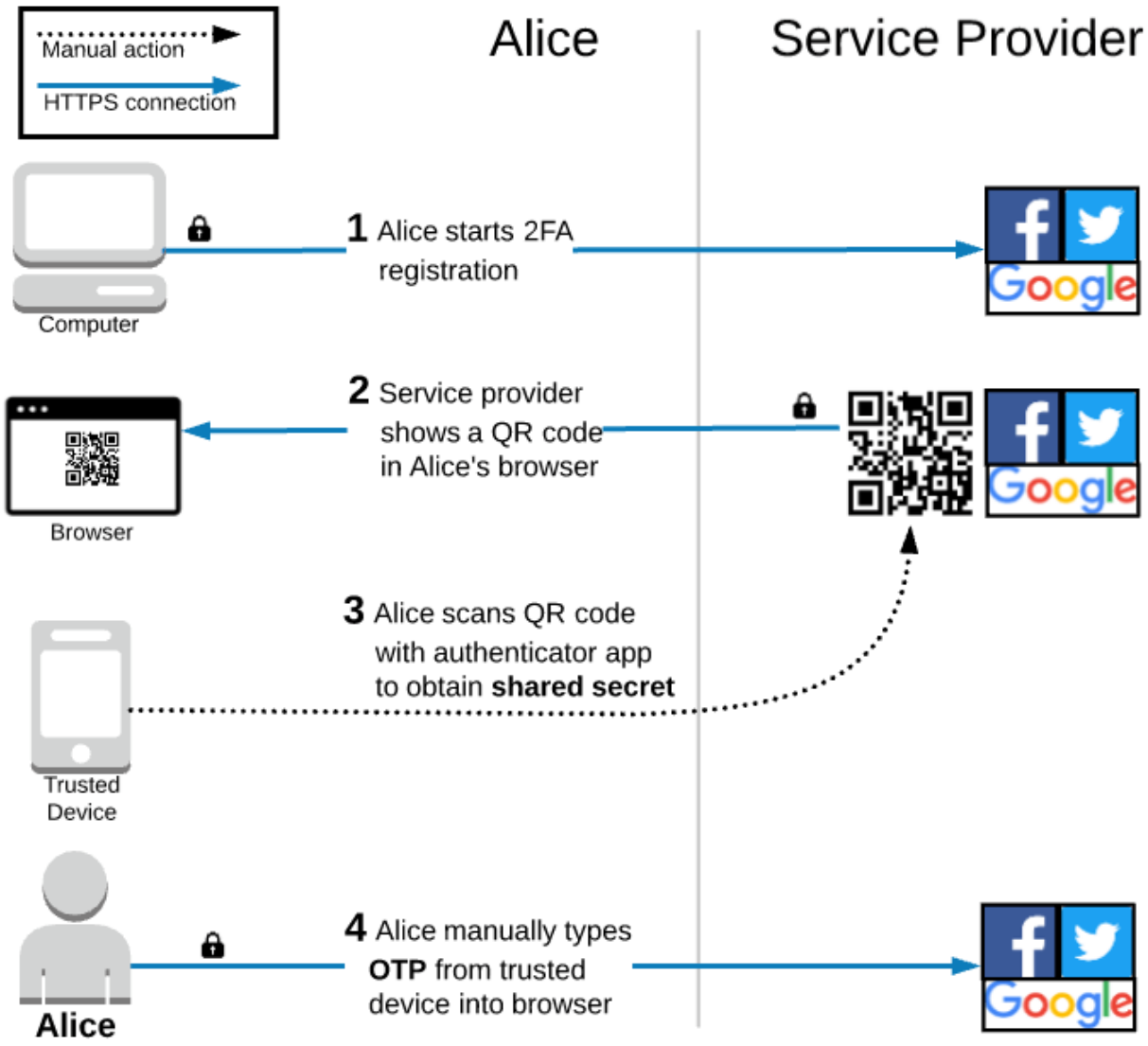

Figure 5 : Exemple de TOTP(Enregistrement)

Le schéma ci-dessus montre le déroulement de l'enregistrement.Si Alice a déjà une application d'authentification installée sur son smartphone. Il lui suffit de scanner le code QR avec son application existante, puis de taper manuellement l'OTP de l'application d'authentification pour activer TOTP 2FA sur son compte.

Alice ne doit effectuer la procédure d'enregistrement qu'une seule fois pour chacun de ses comptes.

Authentification après enregistrement:

La seule chose qu'Alice doit faire pendant le processus de connexion est de taper manuellement dans son navigateur l'OTP affiché sur son appareil de confiance.

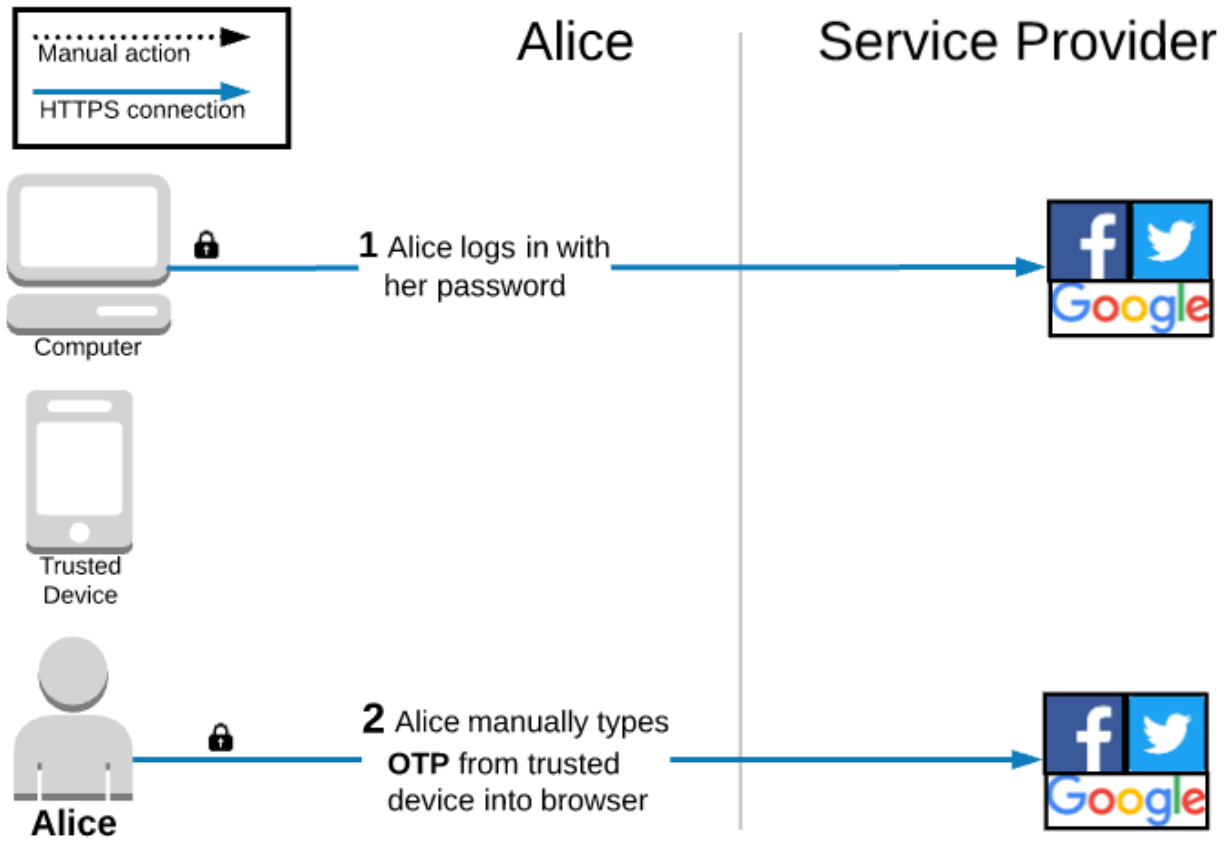

Figure 6 : Exemple de TOTP(Authentification)

La principale amélioration est que rien n'est envoyé sur le réseau du fournisseur de services vers l'appareil d'Alice. Le secret partagé est utilisé pour générer les OTP localement dans l'application d'authentification. Cela signifie qu'Alice n'a pas besoin de connexion de données pour utiliser TOTP 2FA, ce qui est très pratique pour les personnes qui se trouvent dans des régions où la réception est mauvaise, ou qui voyagent à l'étranger où elles n'ont peut-être pas de connexion du tout.

3. Exemples d'utilisation :

Voici quelques exemples d'utilisation de 2FA :

3.1) Code par SMS :

Les grandes compagnies telles que Google, Amazon ou encore Microsoft utilisent le 2FA pour améliorer la sécurité de leurs systèmes, généralement après avoir rentrés leurs mots de passe. Les utilisateurs reçoivent un sms contenant un code (ou token) qu'ils doivent rentrer dans un temps imparti avant que celui-ci ne soit plus valide.

Voici un exemple de **code par SMS** envoyé par Google :

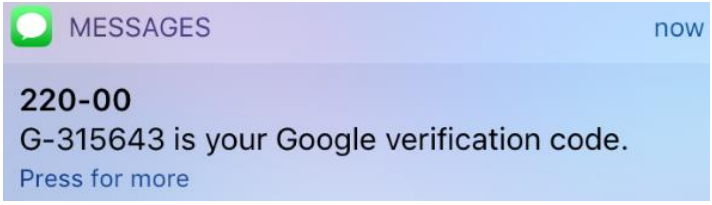

Figure 7 : Exemple d'envoi de code par sms

Ce moyen d'authentification est rapide, voir quasiment instantané, si l'on est dans une zone couverte d'un bon réseau. De plus, elle a l'avantage d'être simple à utiliser et de nos jours presque tout le monde possède un smartphone.

Néanmoins, l'inconvénient de cette méthode est que l'utilisateur doit renseigner son numéro de portable à l'entreprise. Il faut donc faire confiance à celle-ci sur la

gestion des données personnelles. Il faut aussi souligner le fait qu'avec cette méthode on est obligé d'avoir son smartphone sur soi pour se connecter.

Il peut également arriver que le SMS soit intercepté. Il existe des variétés delogiciels malveillants dont l'objectif est d'intercepter le SMS.

3.2) Notifications automatiques :

Les entreprises de jeu vidéo telles que Blizzard, Epic Game ou encore la plateforme Steam, utilisent les notifications automatiques afin de rendre plus robuste les comptes de leurs utilisateurs. C'est un moyen d'authentification qui consiste à envoyer une notification sur un autre appareil (généralement le smartphone) où l'utilisateur a juste à approuver ou refuser la connexion.

Par exemple, Blizzard a créé une application mobile comme support d'authentification à 2 facteurs, qui permet à l'utilisateur de valider une connexion à son compte. Au lieu de rentrer le token généré comme vu précédemment avec l'envoi du token par SMS, l'utilisateur à juste à valider ou non la connexion depuis son mobile.

Figure 8 : Exemple d'une notification automatique

L'énorme avantage de ce procédé est qu'il est très simple et très rapide. Il n'y a pas à recopier un token pour valider la connexion. Par ailleurs, cette méthode est plus sécurisée que l'envoi d'un code par SMS, car il n'y a pas de soucis de phishing.

Cependant, il faut avoir obligatoirement une connexion à internet et son smartphone sur soi pour pouvoir se connecter.

3.3) Clé U2F :

Une autre méthode d'authentification en deux étapes consiste à utiliser des **clés U2F**. Ce sont des appareils physiques qui nous permettent de nous authentifier lors de la connexion à certains services. Ils sont également très utiles pour gagner en sécurité.

Sans aucun doute, l'un des avantages les plus importants est qu'il s'agit d'un appareil physique. Par conséquent, c'est l'une des méthodes les plus sûres car elle est conçue contre les attaques de phishing.

L'un des principaux inconvénients est sa mise en œuvre difficile à de nombreuses reprises. Il s'agit d'une technologie relativement nouvelle et moins répandue. De plus, nous pouvons également avoir une incompatibilité avec certains appareils car il s'agit d'un port USB.

Figure 9 : Exemple de clé U2F

3.4) Biométrie :

D'autre part, nous avons également des méthodes **biométriques** pour nous authentifier. Dans ce cas, nous pouvons utiliser nos empreintes digitales, iris, reconnaissance faciale ou encore notre voix.

Ces données étant propres et uniques à l'utilisateur, cette méthode est très difficile à contrefaire. Il est nettement plus difficile pour un intrus de contrer cette méthode d'authentification (Sauf s'il vous coupe un doigt ..!).

Mais comme toute méthode, il comporte aussi des inconvénients. L'un des principaux est qu'il est difficile à implanter. De plus, de nombreux utilisateurs ne font pas confiance aux sociétés externes qui peuvent utiliser leur voix ou leur image.

3.5) Question de sécurité :

Cette dernière méthode (il en existe d'autres mais on restera sur celle-ci) est sûrement l'option la moins sûre de toutes. Elle consiste simplement à ajouter une **question de sécurité** pour pouvoir accéder à un certain service. Souvent appelé question secrète, le but ici est de demander à l'utilisateur une information qui lui est propre, sa couleur préférée, la marque de sa première voiture, le nom de son premier animal de compagnie etc…

Un avantage est qu'il est très facile à implémenter. De plus, il s'agit simplement d'informations que l'utilisateur connaît par son passé.

Par ailleurs, il a des points négatifs très importants. Quiconque nous connaît très bien ou fait des recherches sur le net pourrait trouver ces réponses et éventuellement voler nos comptes. En outre, ce n'est pas une méthode sûre.

4. Failles et vulnérabilités :

Bien que la 2FA ajoute une couche supplémentaire à la sécurité, cela ne la rend pas invulnérable. Il existe plusieurs approches qu'un hacker peut utiliser pour la contourner :

Piratage de la carte SIM : Dans cette approche, lehacker s'approprie effectivement le numéro de téléphone de l'appareil mobile utilisé dans le cadre de la 2FA. Cela lui permet de recevoir les jetons à usage unique et de se connecter.

Phishing : Le phishing peut diriger les utilisateurs vers des sites malveillants où les mots de passe à usage unique sont saisis. Un pirate qui surveille le site en temps réel peut utiliser le jeton pour accéder au site ciblé avant que le jeton n'expire.

Attaques par force brute : Le principe de cette attaque est de tenter d'entrer par force en essayant une multitude de tentatives possibles afin de trouver la bonne combinaison pour cracker ou deviner un mot de passe.

Pas plus tard que le 24 novembre 2020, un communiqué de presse[2] a été publié par Digital Defense (une société de sécurité informatique américaine) sur une faille de sécurité obtenue par des attaques par force brute, permettant à des cybers attaquants de deviner les paramètres d'URL et de contourner la 2FA.

Enregistreur de frappes: Une possible faille aux 2FA serait d'avoir recours à un logiciel malveillant qui enregistre les frappes entrées sur le clavier de l'ordinateur.Cette méthode permet d'avoir accès aux mots de passe de l'utilisateur ou bien leurs questions secrètes.

5.Implémentation

Pour l'implémentation des algorithmes TOTP et HOTP, il existe de nombreuses librairies. Nous avons choisi Python comme langage de programmation car il dispose de deux librairies qui répondent à ce besoin Pyotp et python-oath.

Implémentation de HOTP:

- **pyotp.random_base32()** permet de générer une clé de 16 caractères
- **pyotp.HOTP(secret_key)** permet d'appliquer l'algorithme HOTP avec la clé secrète générée secret_key
- **x**, **y** et **z** sont respectivement les valeurs des tokens générés aux compteurs 0, 1 et 1401
- **hotp.verify** nous permet de vérifier la validité du token au compteur 1401 et 1402

```python
import pyotp

secret_key = pyotp.random_base32()
hotp = pyotp.HOTP(secret_key)

x = hotp.at(0)
y = hotp.at(1)
z = hotp.at(1401)

print("passcode value at counter 0", x)
print("passcode value at counter 1 ", y)
print("passcode value at counter 1401", z)

# OTP verified with a counter
print("verify passcode validity at counter 1401")
print(hotp.verify(z , 1401))# => True
print("verify passcode validity at counter 1402")
print(hotp.verify(z , 1402)) # => False
```

Figure 10 : Implémentation HOTP

Après exécution du programme nous obtenons le résultat suivant

```
passcode value at counter 0 457439
passcode value at counter 1  538488
passcode value at counter 1401 804157
verify passcode validity at counter 1401
True
verify passcode validity at counter 1402
False
```

Figure 11 : Résultat exécution HOTP

Implémentation de TOTP :

- Comme vu précédemment une clé secrète est générée. On applique cette fois l'algorithme TOTP grâce à la fonction TOTP().
- **totp.now()** nous permet de générer un token calculé à partir du temps actuel.
- On vérifie la validité du token généré passcode.
- On vérifie la validité du passcode 30 secondes après.

```
import pyotp, time

secret_key = pyotp.random_base32()
totp = pyotp.TOTP(secret_key)
passcode = totp.now()

print("verify the passcode for the current time", passcode)
v = totp.verify(passcode)
print(v)
print("waiting for 30 seconds...")
time.sleep(30)
v = totp.verify(passcode)
print(v)
```

Figure 12 : Implémentation TOTP

Après exécution du programme nous obtenons le résultat suivant :

```
verify the passcode for the current time 815013
True
waiting for 30 seconds...
False
```

Figure 13 : Résultat exécution TOTP

6. Conclusion

De nos jours, tout est informatisé.Il est donc de plus en plus important de porter une attention à notre sécurité numérique, particulièrement si l'on veut éviter toute intrusion.

C'est pour cela qu'il émerge de nouvelles méthodes d'identification telles que l'authentification à 2 facteurs, qui permet de mieux sécuriser une connexion en ajoutant une seconde authentification. Elle permet de certifier qu'il s'agit bien du bon utilisateur en vérifiant ces informations via des facteurs qui sont propres et uniques à l'utilisateur tels que des facteurs mémoriels, corporels ou encore matériels.

Avoir recourt au 2FA permet à de nombreuses entreprises de consolider la sécurité des comptes de leurs employés ou de leurs clients. Les employés doivent être formés pour détecter et éviter les courriels de phishing. L'infrastructure doit comprendre des pares-feux, des listes noires, des filtres et d'autres contrôles qui contribuent à protéger les employés et leurs informations d'identification contre les sites dangereux.

Néanmoins, bien que l'on rajoute une seconde étape de vérification, il existe cependant des failles pour contrer cette autre étape d'identification.

C'est pourquoi, pour pallier ces failles, il est possible de cumuler plusieurs facteurs d'authentification à la suite afin de rendre la connexion encore plus sécurisée : on parle ici de MFA (multiple factor authentication).

Finalement, les utilisateurs peuvent se retrouver frustrés de devoir à chaque connexion rentrer tout un lot d'authentification. Il faut donc trouver un juste milieu entre la facilité d'accès de l'utilisateur et la robustesse et sécurité de la connexion.

Webographie

1. Vaadata. Fonctionnement et configuration de l'authentification 2 facteurs. https://www.vaadata.com/blog/fr/fonctionnement-et-configuration-de-lauthentification-2-facteurs/ (2015).

2. Catalin, C. Une faille de sécurité permettant de contourner la 2FA découverte dans le logiciel cPanel - ZDNet. https://www.zdnet.fr/actualites/une-faille-de-securite-permettant-de-contourner-la-2fa-decouverte-dans-le-logiciel-cpanel-39913635.htm (2020).

3. LAMA. Authentification deux facteurs — Wiki du LAMA (UMR 5127). https://www.lama.univ-savoie.fr/mediawiki/index.php/Authentification_deux_facteurs (2018).

4. Maxim Oliynyk. HOTP Algorithm Explained - Protectimus Solutions. https://www.protectimus.com/blog/hotp-algorithm/ (2020).

5. Elliot Williams. Inside Two-Factor Authentication Apps | Hackaday. https://hackaday.com/2017/10/16/inside-two-factor-authentication-apps/#more-277262 (2021).

6. *pyauth/pyotp*. (PyAuth, 2020).https://github.com/pyauth/pyotp

7. Conor Gilsenan. TOTP: (way) more secure than SMS, but more annoying than Push. https://www.allthingsauth.com/2018/04/05/totp-way-more-secure-than-sms-but-more-annoying-than-push/ (2018).

8. Jerry Hildenbrand. Two-factor authentication: Everything you need to know | Android Central. https://www.androidcentral.com/two-factor-authentication (2020).

9. Mills, M. Types de 2FA: avantages et inconvénients | ITIGIC. https://itigic.com/fr/types-of-2fa-advantages-and-disadvantages/ (2020).

10. Alex Chauvin. un OTP avec python | Xin CTO. https://xincto.me/2018/11/un-otp-avec-python.html (2018).