VVT2026

Authentification sans mot de passe

BOUTELIER Sébastien
sebastien.boutelier@lis-lab.fr

Le contexte

Phishing

Compromission de services – vols d’identifiants

Mots de passe oubliés, réutilisés entre plusieurs services, trop simples...

Malware exfiltrant tous les identifiants (mots de passe, tokens, clés SSH…)

Les contre-mesures

- VPN – Peu pratique, problématique quand il faut jongler avec plusieurs VPN.
- Utilisation du MFA – Nécessite toujours d’utiliser un mot de passe et ajoute une étape pénible. Possibilité de harcèlement via certains MFA en cas de vol de mot de passe.

Changement de paradigme

- Utilisation d’un chiffrement fort plutôt qu’un secret partagé (hashcat peut bruteforcer des hash de mots de passe faibles avec des GPU).
- Stocker de manière inviolable les secrets.
- Remplacer ce que l’on sait par ce que l’on a.

Le matériel

- Le TPM (Trusted Platform Module) est une puce passive sur la carte mère permettant des opérations cryptographiques et le stockage sécurisé de secrets.
- La Secure Enclave sur le matériel Apple
- Le TEE (Trusted Execution Environment) sur les processeurs ARM.
- Une clé USB type Yubikey/Nitrokey

Authentification web

Protocole FIDO2 (Fast IDentity Online 2) est une norme ouverte mise au point en 2018 par la FIDO Alliance. Ce protocole est mis en place avec l'API webauthn du W3C au niveau du navigateur et le protocole de communication CTAP avec l'authenticateur.

L'authentification repose sur un couple clé publique/clé privée généré dans un environnement sécurisé. La clé privée ne sortant jamais de cet environnement et ne pouvant pas en être extraite. Il n'y a pas de notion de login ni de mot de passe.

Avantages

- Chiffrement fort par clé publique/clé privée, bruteforce impossible.
- Simple d’utilisation, pas de login ni de mot de passe à connaître, juste un code PIN ou une empreinte digitale. Pas de mot de passe oublié.
- Phishing impossible, une clé privée est liée à un service.
- La compromission d’un fournisseur de service en exposant les clés publiques n’a aucune conséquence. Ces clés publiques ne sont liées qu’à ce service.
- Clés privées stockées de manière sécurisée impossible à extraire sans faille de sécurité bas niveau.

Zero Trust

- La clé privée ne sort jamais de l’authenticateur.
- Chaque session est basée sur une preuve cryptographique forte (algorithme ECDAA basé sur les courbes elliptiques) .
- L’authentification est liée cryptographiquement au domaine rendant le phishing impossible.

Local vs Cloud

Deux types de stockage des clés privées :

- Local sur une puce ou un matériel.
- Cloud dans un compte Google, Apple, une instance bitwarden, un fichier keepass...

Local

Le stockage local :

- Maîtrise de l’environnement, lié à un matériel que l’on possède.
- Il est impossible d’extraire une clé privée, même avec un accès complet à l’appareil.
- Il faut créer une clé d’accès par appareil. Il faut avoir l’appareil avec soi.

Cloud

Stockage dans le cloud :

- La clé privée est stockée de manière chiffrée sur un compte Google ou Apple. Il faut le code PIN d’un des appareils pour déverrouiller sa clé privée et ainsi utiliser la clé d’accès.
- Pirater un compte Google ou Apple ne suffit pas. Il faudra entrer le code PIN sur un des appareils qui a déjà chiffré la clé pour pouvoir ensuite l’utiliser.
- Simplicité car une seule clé fonctionne sur tous les appareils connectés.

L’authentification QR Code

Il est possible de s’authentifier avec un QR Code. Deux prérequis :

- L’appareil lisant le QR Code doit posséder la clé d’accès.
- Les deux appareils doivent avoir du bluetooth activé (mais pas forcément appairé) afin de valider que les deux appareils sont bien proches physiquement.

Processus d’authentification

L’utilisateur se connecte à un service.

- Le service envoie un défi
- Le navigateur reçoit le défi et l’envoie à l’authenticateur qui va le signer numériquement avec la clé privée.
- Le navigateur renvoie le défi signé au fournisseur de service.
- Le fournisseur de service valide la signature avec la clé publique dont il dispose et autorise l’utilisateur si la signature est correcte.

Mise en place

Côté serveur, un serveur SSO supportant les passkeys. Le logiciel keycloak en version 26.4 ou plus :

- Open source
- Authentification OpenID Connect, SAML, CAS en option
- Permet de construire des scénarios d’authentification par service.
- Supporte le TOTP (RFC6238)
- Possibilité de stocker plusieurs clés d’accès par utilisateur.
- Possibilité de créer plusieurs royaumes (un pour les comptes admin, un autre pour les simples mortels)

Client sous windows

Windows supporte nativement les passkeys via windows Hello :

- Les clés sont stockées dans le TPM.
- L’activation de Windows Hello nécessite un code PIN (qui sera utilisé pour l’ouverture de session) ou une empreinte digitale ou de la reconnaissance faciale.
- Fonctionne sur un poste dans un Active Directory (Peut nécessiter une clé dans la base de registre)

Client sous Apple

Le bon élève, la clé peut nativement être stockée dans la Secure Enclave ou dans iCloud.

- Nécessite un code PIN, une empreinte digitale ou la reconnaissance faciale.

Client Linux

Le mauvais élève, ne gère pas nativement les clés d’accès. Il n’existe que deux solutions :

- Une clé physique.
- L’utilisation de Chrome avec un compte Google authentifié pour utiliser le password manager de Chrome et donc la clé d’accès dans le compte Google. Un code PIN sera nécessaire.

Tout n’est pas rose.

Les clés d’accès nécessitent quelques points d’attention :

- Il faut maintenir une procédure d’authentification par mot de passe en cas de perte ou d’indisponibilité des moyens d’authentification.
- Attention aux clés d’accès dans le cloud, les GAFAM s’en servent pour maintenir les utilisateurs captifs d’un écosystème.
- Les clés d’accès sont individuelles donc pas de comptes partagés.
- L’utilisation est simple mais la mise en place peut parfois nécessiter un accompagnement.

Les clés physiques

Pour les admins, les clés physiques ont de nombreux avantages :

- Stockage sécurisé des clés.
- Transportable.
- En cas de vol, nombre d’essais limité pour trouver le code PIN.
- Fonctionne sur tous les OS.
- Peut déverrouiller une base KeepassXC (avec un HMAC), stocker des clés SSH.
- Problème : Nécessite un port USB libre ce qui peut être pénible sur certains PC portables.

Minute pub

Message publicitaire désintéressé. Les clés de la marque Nitrokey sont :

- Européenne ! Société allemande et clés produites en Europe.
- Open source. Le firmware codé en Rust est disponible.

Les clés Yubikey :

- Les plus connues. Support biométrique.
- Ne peuvent pas être flashées. Le firmware ne peut être altéré mais en cas de faille, il faut changer la clé.

Authentification SSH

Pour les postes linux, le TPM peut être utilisé pour stocker des clés SSH :

```
sudo apt install libtpm2-pkcs11-tools libtpm2-pkcs11-1
sudo usermod -a -G tss "$(id -nu)"
tpm2_ptool init
tpm2_ptool addtoken --pid=1 --label=ssh --userpin=MySecretPassword --sopin=MySecretPassword
tpm2_ptool addkey --label=ssh --userpin=MySecretPassword –algorithm=ecc256
mkdir -p ~/.ssh
echo "Host *
   ForwardAgent yes
   User LOGIN
   PKCS11Provider /usr/lib/x86_64-linux-gnu/pkcs11/libtpm2_pkcs11.so" >> ~/.ssh/config
```

Authentification SSH

On récupère la clé publique à insérer dans le LDAP ou le authorized_keys :

```
ssh-keygen -D /usr/lib/x86_64-linux-gnu/pkcs11/libtpm2_pkcs11.so
```

On peut utiliser ssh-agent pour ne pas avoir à entrer son code PIN à chaque SSH :

```
echo "if [[ \`ssh-add -l | grep -c SHA256\` == 0 ]]
then
  ssh-add -s /usr/lib/x86_64-linux-gnu/pkcs11/libtpm2_pkcs11.so
fi" >> ~/.bashrc
```

Chiffrement disque dur

Après installation d'un Linux sur une partition chiffrée, il est possible de stocker une clé de déchiffrement dans le TPM :

```
apt install -y dracut tpm2-tools libtpm2-pkcs11-tools libtpm2-pkcs11-1

systemd-cryptenroll --tpm2-device=auto --tpm2-pcrs=0+7 /dev/nvme0n1p3

echo "add_dracutmodules+=\" tpm2-tss crypt \"" >> /etc/dracut.conf

sed -i 's/GRUB_CMDLINE_LINUX=""/GRUB_CMDLINE_LINUX="rd.auto rd.luks=1"/g' /etc/default/grub

cat /dev/null > /etc/crypttab

dracut -f

update-grub
```

ATTENTION !! Le Grub doit être protégé pour éviter le init=/bin/bash

Les limites

Il reste encore quelques problèmes à résoudre :

- Le SMTP, protocole souvent utilisé après un vol d’identifiants.
- Le vol de cookie de sessions.

Le SMTP

Solutions :

- VPN mais pénible pour les utilisateurs
- Tout bloquer sauf les AS des opérateurs téléphoniques.
- Mise en liste blanche temporaire des IP ayant réussi une authentification forte.

Le vol de cookie de session

Le DBSC (Device Bound Session Credentials) proposé par Google est en cours de normalisation au W3C

- Une modification du flux de connexion est faite pour ajouter une entête Secure-Session-Registration.
- Ajout de points de terminaison d’enregistrement et de renouvellement de session
- Cookie de session de courte durée
- Spécifications : https://w3c.github.io/webappsec-dbsc/

Principe du DBSC

- Après authentification, le navigateur demande une clé publique au TPM et l'envoi au serveur distant ; ce qui démarre une DBSC Session et génère un cookie de session à courte durée de vie.
- Quand le cookie expire, le navigateur envoie une demande d'actualisation, le serveur répond avec un défi.
- Le navigateur transmet le défi au TPM qui le signe et le navigateur renvoie le défi signé au serveur.
- Le cookie de session est rafraichi si le serveur valide le défi signé.

C’est la fin...

Remerciements et questions.