Journal : Les bases de l'authentification, clé de sécurité FIDO2 sous Linux et Windows

23 nov. 2025 0

Posté par **Sébastien Rohaut** le 23 novembre 2025 à 19:43.

Étiquettes : **linux**, **sécurité**, **windows**, **authentification**, **double_authentification**, **fido**, **fido2**

Sommaire

> **NOTE**
>
> Cet article est aussi posté sur mon petit blog personnel

Les grands principes de l'authentification

Quelle est la différence entre identification et authentification ? Qu'est-ce que l'authentification multifacteur et pourquoi c'est important ? Comment utiliser une clé de sécurité physique sous Linux et Windows ?

l'identité

Quand on se connecte à un service, on doit s'identifier.

S'**identifier**, c'est qui on est (ou pour qui en souhaite se faire passer). Par exemple, vos prénoms et nom, votre pseudo, votre identifiant pour vous connecter à votre machine au bureau, votre adresse email, etc.

L'authentification

S'**authentifier**, c'est justifier de son identité, prouver qu'on est celui qu'on prétend être.

Pour se présenter dans la vie quotidienne, l'identité (je suis M. Untel), ça suffit généralement. Mais lors d'un contrôle de police, par exemple, on doit justifier son identité, par tous moyens: carte d'identité, passeport, permis de conduite (le tout peut être dématérialisé), etc. Sur un ordinateur, pour prouver que vous êtes bien celui que vous prétendez être, vous rentrez généralement un mot de passe. Parfois, ce peut être aussi votre empreinte digitale sur un capteur du laptop ou du smartphone, ou votre visage, ou votre œil, ou la saisie d'un code additionnel, et ainsi de suite.

Les trois grandes méthodes d'authentification

Il y a trois grands moyens de prouver qui vous êtes:
* **Quelque chose que vous connaissez** (*something you know*) : par exemple un mot de passe (password), une phrase (passphrase), un code PIN, etc.
* **Quelque chose que vous possédez** (*something you have*) : une clé (physique ou virtuelle), un smartphone (via une application OTP par exemple, ou un capteur NFC), un badge, etc.
* **Quelque chose que vous êtes - Qui vous êtes** (*something you are*) : vos empreintes digitales, votre visage, votre iris, etc. : c'est authentification biométrique.

Ces moyens ne sont pas exclusifs, et peuvent être cumulatifs.

L'authentification multifacteur

Dans de très nombreux cas, comme la connexion sur un ordinateur personnel, on utilise un mot de passe. Mais de plus en plus souvent, et à juste raison, il nous est demandé d'ajouter une nouvelle méthode pour s'authentifier, comme un code reçu par SMS ou par email, ou une valeur numérique fournie par une application comme Google Authenticator, Microsoft Authenticator, PingID. Le fait de demander plus d'une seule méthode d'authentification s'appelle le **MFA**, *Multi Factor Authentication*, ou **Authentification multifacteur**. Tout simplement parce que, par exemple, votre mot de passe peut avoir fuité quelque part, mais si la personne qui tente de l'utiliser à votre place n'a pas accès à votre smartphone, ou vos emails, elle n'obtiendra pas le second facteur d'authentification, et ne pourra pas accéder aux services qu'elle tente d'utiliser: elle n'arrivera pas à usurper votre identité.

Vous entendrez parler aussi de **2FA**, c'est le cas le plus courant où il vous est demandé deux méthodes, généralement un mot de passe, et un code généré ou reçu par sms ou email.

Exemple de SSH

Un protocole sécurisé

Pour se connecter sur un ordinateur distant, via un réseau (privé, ou public), on utilise un protocole sécurisé: la connexion est codée (chiffrée), et on doit décliner son identité et s'authentifier. La plupart des systèmes d'exploitation (Unix - Linux, MacOS, mais aussi Windows) peuvent utiliser **SSH** *Secure Shell*. Windows peut aussi utiliser **RDP** *Remote Desktop Protocol* pour l'environnement graphique.

Les clés et le chiffrement asymétrique

Le serveur SSH (la machine sur laquelle on souhaite se connecter) peut accepter plusieurs méthodes d'authentification. La plus connue est le mot de passe: vous entrez votre identifiant, puis votre mot de passe (c'est le client qui demande le mot de passe, pas le serveur), qui est ensuite chiffré, envoyé au serveur et comparé avec celui qui y est stocké. S'ils correspondent, alors vous êtes autorisés à vous connecter. Vous venez d'utiliser une authentification de type "*quelque chose que vous connaissez*".

Une autre méthode consiste à utiliser des clés cryptographiques, en fait une **clé privée** et une **clé publique**. On abordera ici cette notion de manière très élémentaire, mais la clé privée ne doit jamais quitter votre ordinateur (ou votre smartphone), et ne doit jamais être divulguée. La clé publique, elle, peut être déployée partout ou vous souhaitez vous connecter, par exemple sur le compte de la machine sur laquelle vous souhaitez vous connecter. Quand vous vous connectez, vous précisez une clé privée que vous souhaitez utiliser. La machine distante va chiffrer un message avec votre clé publique et vous l'envoyer. Seule la clé privée associée à cette clé publique peut décoder ce message. Même la clé publique ne peut pas décoder le message qu'elle a encodé. Durant cette phase, appelée le challenge (il existe une étape où le client doit répondre à un challenge mathématique basé sur les clés), le serveur envoie au client sa clé publique, que le client utilisera pour envoyer sa réponse. Si tout correspond (le serveur a bien reçu ce qui était attendu lors du challenge), ils se partageront une clé de session utilisée cette fois pour un chiffrement symétrique.

C'est ce qu'on appelle de la cryptographie asymétrique, vous trouverez plus de détails sur la page Wikipedia [W] qui décrit tout ceci très bien, ainsi que cette page sur le challenge SSH qui décrit ce qui se passe lors d'une connexion SSH.

SSH et les trois méthodes d'authentification

Lorsque vous utilisez une clé privée pour vous connecter, vous utilisez une méthode "*quelque vous que vous possédez*". Vous possédez cette clé, même si elle est virtuelle.

Lorsque vous générez cette clé, vous pouvez encoder votre clé privée avec une phrase (passphrase). C'est "*quelque chose" que vous connaissez*".

Il est aussi possible d'utiliser des méthodes comme l'empreinte digitale, et dans ce cas, c'est "*quelque chose que vous êtes*". Nous ne l'aborderons pas ici (je n'ai pas que quoi tester).

Les clés physiques

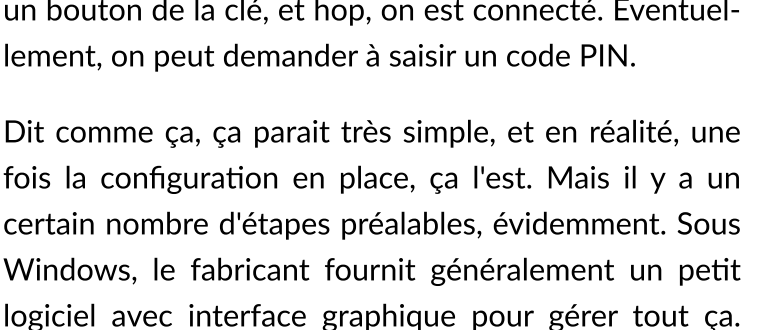
La clé FIDO2 / U2F

Vous avez probablement déjà vu, dans des vidéos, ou des films, ou ailleurs, des gens utiliser des clés USB pour se connecter sur un ordinateur. Ce n'est pas un mythe, ça existe vraiment, et j'en possède une. Sans l'utiliser quotidiennement, elle me permet de faire des tests variés, sous Linux, et je peux vous assurer que ça fonctionne très bien.

Mon modèle est la Thetis PRO-A. Cette clé de sécurité a plusieurs fonctionnalités:
* Compatibilité **FIDO2** *Fast IDentity Online* / **U2F** *Universal Second Factor* avec de la place pour 200 clés.
* **TOTP** *Time Based On Time Password* et **HOTP** *HMAC One Time Password*
* **NFC** *Near-Field Communication*, comme sur les smartphones, cartes de paiement et badges par exemple.

La clé permet une authentification sans mot de passe. En gros, on insère la clé, on se connecte au service souhaité, ce service attend une action, on appuie sur un bouton de la clé, et hop, on est connecté. Éventuellement, on peut demander à saisir un code PIN.

Dit comme ça, ça parait très simple, et en réalité, une fois la configuration en place, ça l'est. Mais il y a un certain nombre d'étapes préalables, évidemment. Sous Windows, le fabricant fournit généralement un petit logiciel avec interface graphique pour gérer tout ça. Sous Linux, c'est souvent en ligne de commande (il existe des interfaces comme fido2-manage).



Je vous propose de voir comment on fait tout ça en ligne de commande. Parce que c'est plus drôle, non ?

Détection sous Linux

FIDO et U2F sont des normes ouvertes, ce qui signifie qu'elles sont librement accessibles et implémentables, et que tout matériel (les clés) respectant ces normes est compatible avec tout produit les supportant. Donc, une même clé peut être utilisée sur tout système d'exploitation, ou tout logiciel, implémentant les normes FIDO. C'est le cas de Linux.

En branchant la clé sur un port USB, voici ce qui se passe au niveau système et noyau. La clé est détectée, notamment en tant que périphérique **HID** *Human Interface Device*, car il faudra interagir avec elle :

```
[  379.375639] usb 1-3: new full-speed USB dev
[  379.500250] usb 1-3: New USB device found,
[  379.500278] usb 1-3: New USB device strings
[  379.500286] usb 1-3: Product: Security Key(
[  379.500293] usb 1-3: Manufacturer: Thetis
[  379.636562] input: Thetis Security Key(FE25
[  379.726606] hid-generic 0003:1EA8:FE25.0002
[  379.727981] hid-generic 0003:1EA8:FE25.0003
[  379.728058] usbcore: registered new interfa
[  379.728061] usbhid: USB HID core driver
```

La clé est bien listée comme périphérique USB:

```
seb@Y13:~$ lsusb
Bus 001 Device 007: ID 1ea8:fe25 Thetis Security Key(FE25)
```

Sur mon installation Ubuntu / Mint (identique au niveau système), des périphériques HID et Input sont créés pour communiquer avec la clé:

```
seb@Y13:~$ ls -l /dev/usb/hiddev0
crw------- 1 root root 180, 0 nov.  22 18:28 /
root@Y13:/dev# ls -l hidraw{1,2}
crw-------  1 root root 241, 1 nov.  23 17:25
crw-rw----+ 1 root root 241, 2 nov.  23 17:25
root@Y13:/dev# ls -l input/by-id/usb-Thetis_Se
lrwxrwxrwx 1 root root 10 nov.  23 17:25 input
```

Les détail sont intéressants, puisqu'une partie du périphérique est d'ailleurs détectée comme un clavier. Logique, puisqu'il y a un bouton physique sur la clé, donc, si vous voulez, une touche de clavier. Mais aussi Chip/SmartCard, puisqu'on va pouvoir écrire dessus pour y stocker des clés.

```
Bus 001 Device 007: ID 1ea8:fe25 Thetis Securi
Device Descriptor:
  iManufacturer           1 Thetis
  iProduct                2 Security Key(FE25)
  Configuration Descriptor:
    Interface Descriptor:
      bInterfaceClass         3 Human Interfac
      bInterfaceSubClass      1 Boot Interface
      bInterfaceProtocol      1 Keyboard
    Interface Descriptor:
      bInterfaceClass        11 Chip/SmartCard
      bInterfaceSubClass      0 [unknown]
```

Installation des commandes

Il faut installer quelques paquetages pour disposer des commandes et bibliothèques nous permettant d'utiliser notre clé:
* Les outils FIDO2 pour gérer la clé
* Le module PAM U2F pour gérer l'authentification associée
* L'outil de configuration du module PAM U2F

```
seb@Y13:~$ sudo apt install fido2-tools pamu2f
```

Pour vérifier si la clé FIDO2 est bien reconnue:

```
seb@Y13:~$ fido2-token -L
/dev/hidraw2: vendor=0x1ea8, product=0xfe25 (T
```

Vous devez associer un **code PIN** à votre clé. Ce code vous sera demandé lors de certaines actions d'administration, ou lorsque vous tenterez de vous connecter si vous le précisez. Si la clé n'a pas encore de code PIN:

```
seb@Y13:~$ fido2-token -S /dev/hidraw2
Enter new PIN for /dev/hidraw2:
Enter the same PIN again:
```

Si vous souhaitez changer de code PIN:

```
seb@Y13:~$ fido2-token -C /dev/hidraw2
Enter current PIN for /dev/hidraw2:
Enter new PIN for /dev/hidraw2:
Enter the same PIN again:
```

Lier sa clé à son compte

Chaque clé physique est personnelle. Chaque clé dispose d'une signature. Il faut l'associer à votre compte Linux, l'enregistrer, en la plaçant dans un fichier. Si vous avez plusieurs clés, elles doivent être ajoutées sur la même ligne. Cette clé n'est pas un secret en tant que tel: elle ne peut pas être réutilisée depuis une autre clé, puisque générée dynamiquement au moment de l'exécution de la commande. Si vous relancez la commande, vous en obtiendrez une autre. Vous devez cependant vous assurer que personne d'autre que vous ne peut modifier votre fichier (644), ou, tout simplement, interdire l'accès à tous sauf à vous (PAM étant root, il pourra lire votre fichier).

```
seb@Y13:~$ mkdir -p ~/.config/fido2
seb@Y13:~$ pamu2fcfg --pin-verification > ~/.c
chmod 600 ~/.config/fido2/u2f_keys
```

On obtient par exemple:

```
seb:6fIc9zAq...BymHRw==,es256,+presence+pin
```

Petite explication rapide sur des deux dernières valeurs
* **presence**: affiche le message *please touch the device.*
* **pin**: demande par défaut le code PIN lors d'une authentification

Si on ne veut pas le code PIN, alors on retire +pin, ça demandera uniquement de toucher le bouton de la clé.

Premier test avec sudo

> [!CAUTION]
>
> Toujours ouvrir une ou plusieurs sessions en root avant de modifier les fichiers PAM, et les sauvegarder avant modification. Ça pourrait être l'unique moyen de vous en sortir en cas d'erreur.

On va maintenant tester avec **sudo**. Pour ce premier test, on va rendre optionnel l'utilisation de la clé. Si elle est présente, elle sera demandée, mais même en cas d'absence ou d'erreur, le mot de passe sera demandé. Il nous faut modifier la configuration **PAM** *Pluggable Authentication Module*, qui gère tous les mécanismes d'authentification sous Linux.

Dans **/etc/pam.d/sudo**:

L'ordre des lignes est important. Notamment pour les lignes **auth** liées à l'authentification. Elles sont lues et traitées dans leur ordre d'apparition. Et la première validée (authentification OK) peut suffire même si d'autres sont présentes. Ici, si on veut tout de même la saisie du mot de passe, on place la ligne après *@include common-auth*, Sinon, on la place avant.

```
auth    sufficient   pam_u2f.so nouserok authf
@include common-auth
```

Pour rendre obligatoire l'utilisation de la clé, remplacer *sufficient* par *
required.

```
auth    required    pam_u2f.so   cue authfile=
```

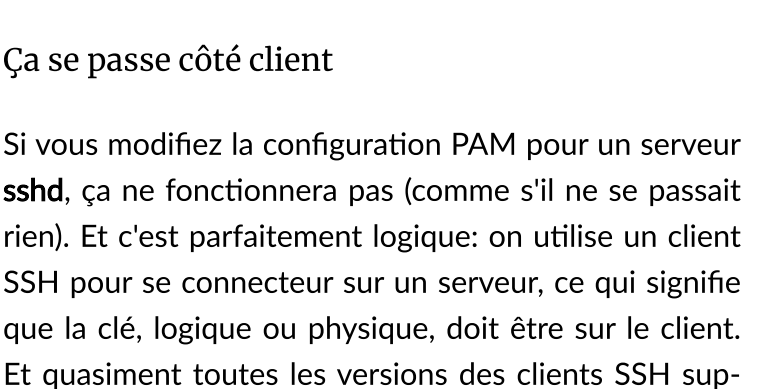
Quelques détails sur les options:
* **cue**: affiche le message demandant de cliquer
* **authfile**: chemin absolu, ou relatif au répertoire personnel
* **nouserok** : autorise les utilisateurs qui n'ont pas de clé (s'ils l'ont ce sera demandé, sinon ça passe à la méthode d'authentification suivante).

test avec LightDM

Si ça fonctionne, on peut étendre si on le souhaite à tout type de login en modifiant le fichier **common-auth**. On peut aussi modifier le tout pour ne pas demander le mot de passe et ne conserver que la clé. Certains **DM** *Display Managers* supportent l'utilisation des clés FIDO2 pour les sessions graphiques. C'est le DM qui affiche l'écran d'identification et d'authentification avant de pouvoir ouvrir sa session graphique. Par exemple, LightDM, par défaut sous Linux Mint, fonctionne bien avec la clé.

```
root@Y13:/etc/pam.d# cat lightdm
#%PAM-1.0
auth    sufficient   pam_u2f.so nouserok authf
```

À la connexion, mon code PIN est demandé, puis je dois appuyer sur le bouton de la clé. J'ai un petit bug ici où le message de demandant d'appuyer sur la clé est affiché après avoir appuyé dessus, mais je peux ensuite me connecter. Si la clé n'est pas insérée alors LightDM me demande mon mot de passe. Je m'excuse pour la qualité de la photo.



Le cas de SSH

Ça se passe côté client

Si vous modifiez la configuration PAM pour un serveur **sshd**, ça ne fonctionnera pas (comme s'il ne se passait rien). Et c'est parfaitement logique: on utilise un client SSH pour se connecteur sur un serveur, ce qui signifie que la clé, logique ou physique, doit être sur le client. Et quasiment toutes les versions des clients SSH supportent les clés FIDO2 depuis des années (merci Yubikey), quel que soit le système d'exploitation.

L'astuce consiste à générer une clé SSH côté client qui sera stockée dans la clé physioque FIDO2, avec ou sans passphrase, avec ou sans validation par code PIN. C'est donc sur le client que la clé doit être connectée.

```
seb@Y13:~$ ssh-keygen -t ed25519-sk -O resider
```

- **Resident** : la clé est résidente, stockée sur la clé FIDO2. Le code PIN sera demandé pour l'y stocker. Cette clé pourra être utilisée partout où la clé FIDO2 est reconnue (Linux, Windows, MacOS, ...). Une référence sera aussi copiée sur disque (~/.ssh/) mais sera inutile sans la clé FIDO2.
- **verify-required** : le code PIN sera demandé.

Test sous Windows

En entreprise, notamment dans les très grosses, les postes de travail sont normalisés, et bien généralement, Windows reste la règle (rassurez-vous, quand c'est bien fait ça passe très bien). Nous allons donc tenter d'utiliser la clé, pourtant configurée sous Linux, sous Windows. FIDO2 est une norme ouverte, n'oubliez pas !

Comme expliqué ci-dessus, on énéère une clé SSH qui sera stockée sur la clé FIDO2, depuis Powershell (oui, j'aime Powershell):

```
PS C:\Users\sebas> ssh-keygen -t ed25519-sk -C
Generating public/private ed25519-sk key pair.
You may need to touch your authenticator to au
A resident key scoped to 'ssh:' with user id '
Overwrite key in token (y/n)? y
You may need to touch your authenticator again
Enter file in which to save the key (C:\Users\
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users
Your public key has been saved in C:\Users\seb
The key fingerprint is:
SHA256:SomBmYnO667vvFXZHQvQdgz+HwqhFfmE7lHbbY6
The key's randomart image is:
+[ED25519-SK 256]-+
|      ..o=       |
| . =   o=.=      |
|. = .  o==.o .   |
|o    o *o=oo. o  |
| o  . *.S.+ .+   |
|  .  o ... o...  |
| .  . .   . ..   |
|.. .       o  .  |
|+==.      E .oo. |
+----[SHA256]-----+
```

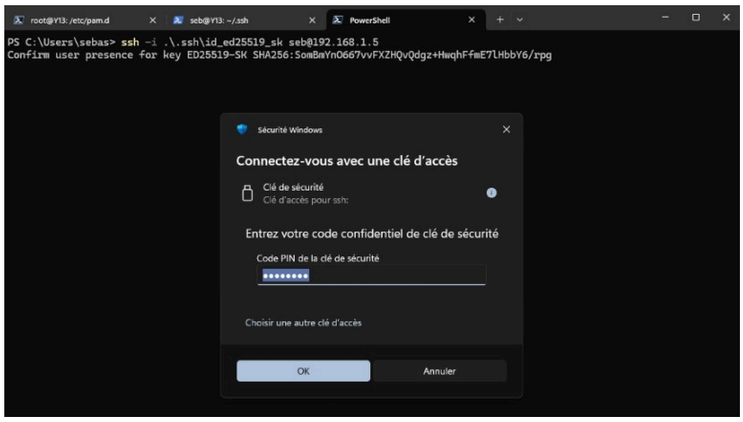
On obtient, comme sous Linux, deux fichiers dans le *.ssh*.

```
PS C:\Users\sebas\.ssh> dir


    Directory: C:\Users\sebas\.ssh


Mode                 LastWriteTime         Len
----                 -------------         ---
-a---          23/11/2025    14:20
-a---          23/11/2025    14:20
-a---          22/11/2025    18:46           1
```

Pour rappel, la clé privée est une référence pour indiquer d'utiliser la clé stockée sur la clé physique FIDO2. Seule, elle ne fonctionne pas, car le système d'exploitation demandera de connecter la clé physique pour accéder à la clé qui y est stockée.

On recopie ensuite, comme toujours, la clé publique dans le authorized_keys de la destination. Puis on teste:

```
PS C:\Users\sebas> ssh -i .\.ssh\id_ed25519_sk
Confirm user presence for key ED25519-SK SHA25
User presence confirmed


Last login: Sun Nov 23 14:32:05 2025 from 192.
seb@Y13:~$
```



Pas mal, hein ?

Le mot de la fin

On voit clairement la pertinence de l'existence des clés physiques, l'utilité qu'elles peuvent avoir, et finalement la facilité utilisation une fois la clé et le système configurés. Ces clés peuvent avoir d'autres usages, notamment pour se connecter à de nombreux services, sur le web ou non. Ainsi, de nombreux développeurs s'en servent pour pousser leur code sur les dépôts Github, et d'autres utilisateurs pour des services comme gmail, par exemple, avec une authentification intégrée aux navigateurs web. Je vous souhaite beaucoup d'amusement.

(0 commentaire). Markdown EPUB

Envoyer un commentaire

Suivre le flux des commentaires

Note : les commentaires appartiennent à celles et ceux qui les ont postés. Nous n'en sommes pas responsables.

Revenir en haut de page