Universial 2nd Factor (U2F)

Universal 2nd Factor (U2F) is an open standard that strengthens and simplifies two-factor authentication (2FA) using specialized USB or NFC devices based on similar security technology found in smart cards.

For support of U2F in major web browsers and system authentication install the following packages:

```
pacman -S libfido2 pam-u2f
```

Generate U2F key for PAM

> **NOTE:** Generate keys as a regular user!

To start using a U2F key for system-level authentication, keys need to be created first.

The default directory these keys will usually be looked for is at `~/.config/Yubico` (since the `pam-u2f` package is developed by Yubico for use with their Yubikeys, but it works with other keys as well).

Create the directory under your home directory:

```
mkdir ~/.config/Yubico
```

The `pam-u2f` package comes with a utility to create keys from the USB device. Create new keys with `pamu2fcfg`:

> **WARNING:** This takes your machine's current host name and assumes it is not re-assigned on network changes! Changing your machine's host name might render the key unable to authenticate you until your machine returns to the original host name.

> **NOTE:** Keep an eye on your hardware security token, as it might silently indicate it is waiting on user interaction to continue.

```
pamu2fcfg -o pam://$HOST -i pam://$HOST > ~/.config/Yubico/u2f_keys
```

System-wide U2F prompts

ATTENTION: A potentially undesirable side effect of this method is that any keychains that use the user password to unlock, such as the login keychain in GNOME or KDE, will immediately request the password after login. Since this allows passwordless logins, the user password for unlocking will not be passed on to the secrets provider. If you depend on automatic unlocking of the login keychain, e.g. for SSH key passphrases or Wi-Fi passwords, see one of the other methods below.

To use your physical security key system-wide and not just for specific use-cases, add the following line **before** the first `auth` line in `/etc/pam.d/system-auth`:

NOTE: Be sure to replace `hostname` with the actual host name of your machine!

```
auth            sufficient      pam_u2f.so cue origin=pam://hostname appid=pam://hostname
```

This will prompt you to touch your physical security key during every attempt at authenticating with your user, whether it's in conjunction with graphical system administrator prompts, `sudo` prompts, display manager login prompts, TTY logins, etc.

If the security key is not connected, the system will fall back to regular password prompts.

Passwordless `sudo`

WARNING: Changes to PAM configuration files apply immediately! Before making any changes to your configuration, start a separate shell with root permissions (e.g. `sudo -s`). This way you can revert any changes if something goes wrong.

A U2F key can be set up for `sudo` to allow for passwordless system maintenance tasks in the terminal.

Open `/etc/pam.d/sudo` and add the following line **before** the first `auth` line:

NOTE: Be sure to replace `hostname` with the actual host name of your machine!

```
auth            sufficient      pam_u2f.so cue origin=pam://hostname appid=pam://hostname
```

To test, open a new terminal and type `sudo ls`. Your key's LED should flash and after clicking it the command is executed. The option `cue` causes an instruction to appear on what to do, e.g. `Please touch the device`.

Note that setting this does not include graphical prompts to elevate privileges in desktop environment such as GNOME or KDE. See the following section for these types of use cases.

Passwordless Polkit

Many graphical applications rely on Polkit to elevate privileges. Polkit can be set up for passwordless authentication in much of the same way as `sudo`.

By default, there is no Polkit PAM configuration present. To add it, copy the default configuration file that comes with Polkit into the PAM system configuration directory:

```
sudo cp /usr/lib/pam.d/polkit-1 /etc/pam.d/polkit-1
```

Then edit `/etc/pam.d/polkit-1`, adding the following line **before** the first `auth` line in the file:

> **NOTE:** Be sure to replace `hostname` with the actual host name of your machine!

```
auth       sufficient   pam_u2f.so cue origin=pam://hostname appid=pam://hostname
```

2nd factor in GDM

A U2F key can be used in addition to your password for added security.

Open `/etc/pam.d/gdm-password` and add the following line **after** the existing `auth` lines:

> **NOTE:** Be sure to replace `hostname` with the actual host name of your machine!

```
auth          required     pam_u2f.so nouserok cue origin=pam://hostname appid=pam://hostname
```

This will require you to have your U2F physical key inserted to authenticate and log you in with your local user account.

> **WARNING:** If you lose your key you will also lose your ability to authenticate and log in to your user account. You could theoretically use `sufficient` instead of `required` but this would

render the security benefits of this endeavour pointless, as the password would still be enough to gain access to your account.

Please note the use of the `nouserok` option which allows the rule to fail if the user did not configure a key or the key is not connected. The `cue` option will display a prompt to let you know the physical key is waiting for you to touch it.

Unlock LUKS container during boot

A FIDO2 key can also be used to unlock your LUKS encrypted drives. To register the key, you will need to use the `systemd-cryptenroll` utility and have a systemd-based initrd.

Run the following command to list your detected keys:

```
systemd-cryptenroll --fido2-device=list
```

Then you can register the key in a LUKS slot, specifying the path to the FIDO2 device, or using the `auto` value if there is only one device:

ATTENTION: Make sure to pass the device node of your actual LUKS container!

```
systemd-cryptenroll --fido2-device=auto /dev/nvme0n1p2
```

To make systemd use the FIDO2 key for unlocking during boot, add the following option to your `rd.luks.options` list of options:

```
rd.luks.options=fido2-device=auto
```

Alternatively, if you do not want to pass this as a kernel command line option, add the option to your `/etc/crypttab.initramfs` and regenerate your initramfs after you've made changes:

```
# <name>	<device>									<passphrase>	<options>
root		UUID=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX		none			fido2-device=auto
```

When booting your system, watch for the indicator on your FIDO2 hardware key prompting you to touch it.

Revision #6
Created 2 March 2022 20:55:22 by Sebin
Updated 1 March 2025 02:40:08 by Sebin