systemd-cryptenroll

Related articles

- dm-crypt
- Smartcards
- Universal 2nd Factor
- Trusted Platform Module
- Unified Extensible Firmware Interface/Secure Boot

From systemd-cryptenroll(1) (https://man.archlinux.org/man/systemd-cryptenroll.1):

> systemd-cryptenroll is a tool for enrolling hardware security tokens and devices into a LUKS2 encrypted volume, which may then be used to unlock the volume during boot.

systemd-cryptenroll allows enrolling **smartcards**, **FIDO2** tokens and **Trusted Platform Module** security chips into **LUKS** devices, as well as regular passphrases. These devices are later unlocked by `systemd-cryptsetup@.service(8) (https://man.archlinux.org/man/systemd-cryptsetup%40.service.8)`, using the enrolled tokens.

1 Installation

systemd-cryptenroll is part of and packaged with `systemd (https://archlinux.org/packages/?name=systemd)`. However, extra packages are required to use hardware devices as keys:

- To use PKCS#11 tokens, **install** `libp11-kit (https://archlinux.org/packages/?name=libp11-kit)`, you may also need `opensc (https://archlinux.org/packages/?name=opensc)` and `opensc-p11-kit-module (https://aur.archlinux.org/packages/opensc-p11-kit-module/)`[AUR].
- To use FIDO2 tokens, install `libfido2 (https://archlinux.org/packages/?name=libfido2)`.
- To use TPM2 devices, install `tpm2-tss (https://archlinux.org/packages/?name=tpm2-tss)`.

2 List keyslots

systemd-cryptenroll can list the keyslots in a LUKS device, similar to `cryptsetup luksDump`, but in a more user-friendly format.

```
# systemd-cryptenroll /dev/disk
```

```
SLOT TYPE
   0 password
   1 tpm2
```

3 Erasing keyslots

```
# systemd-cryptenroll /dev/disk --wipe-slot=SLOT
```

Where *SLOT* can be:

- A single keyslot index, as represented in **#List keyslots**
- A type of keyslot, which will erase all keyslots of that type. Valid types are `empty`, `password`,

`recovery`, `pkcs11`, `fido2`, `tpm2`
- A combination of all of the above, separated by commas
- The string `all`, which erases all keyslots on the device. This option can only be used when enrolling another device or passphrase at the same time.

The `--wipe-slot` operation can be used in combination with all enrollment options, which is useful to update existing device enrollments:

```
# systemd-cryptenroll /dev/disk --wipe-slot=fido2 --fido2-device=auto
```

4 Enrolling passphrases

4.1 Regular password

This is equivalent to `cryptsetup luksAddKey`.

```
# systemd-cryptenroll /dev/disk --password
```

4.2 Recovery key

From **systemd-cryptenroll(1) (https://man.archlinux.org/man/systemd-cryptenroll.1)**:

> Recovery keys are mostly identical to passphrases, but are computer-generated instead of being chosen by a human, and thus have a guaranteed high entropy. The key uses a character set that is easy to type in, and may be scanned off screen via a QR code.

A recovery key is designed to be used as a fallback if the hardware tokens are unavailable, and can be used in place of regular passphrases whenever they are required.

```
# systemd-cryptenroll /dev/disk --recovery-key
```

5 Enrolling hardware devices

The `--type-device` options must point to a valid device path of their respective type. A list of available devices can be obtained by passing the `list` argument to this option. Alternatively, if you only have a single device of the desired type connected, the `auto` option can be used to automatically select it.

> **Note:** After enrolling the hardware tokens into the LUKS2 volumes, you must configure your system to use them when appropriate. See **dm-crypt/System configuration#Trusted Platform Module and FIDO2 keys** for volumes that should be unlocked in early userspace like the root filesystem, and **dm-crypt/System configuration#Unlocking in late userspace** for other partitions.

5.1 PKCS#11 tokens or smartcards

The token or smartcard must contain a RSA key pair, which will be used to encrypt the generated key that will be used to unlock the volume.

```
# systemd-cryptenroll /dev/disk --pkcs11-token-uri=device
```

5.2 FIDO2 tokens

Any FIDO2 token that supports the "hmac-secret" extension can be used with *systemd-cryptenroll*. The following example would enroll a FIDO2 token to an encrypted LUKS2 block device, requiring only user presence as authentication.

```
# systemd-cryptenroll /dev/disk --fido2-device=device --fido2-with-client-pin=no
```

In addition, *systemd-cryptenroll* supports using the token's built-in user verification methods:

- `--fido2-with-user-presence` defines whether to verify the user presence (i.e. by tapping the token) before unlocking, defaults to yes
- `--fido2-with-user-verification` defines whether to require user verification before unlocking, defaults to no

Note:

- These options will have no effect if the token does not support these features.
- See User Presence vs User Verification (https://developers.yubico.com/WebAuthn/WebAuthn_Developer_Guide/User_Presence_vs_User_Verification.html) for more information on the difference between the two.

By default, the cryptographic algorithm used when generating a FIDO2 credential is *es256* which denotes Elliptic Curve Digital Signature Algorithm (ECDSA) over NIST P-256 with SHA-256. If desired and provided by the FIDO2 token, a different cryptographic algorithm can be specified during enrollment.

Note: This may also be desirable for those concerned with ECDSA. See SSH keys#ECDSA for details.

Suppose that a previous FIDO2 token has already been enrolled and the user wishes to enroll another, the following generates an *eddsa* credential which denotes EdDSA (https://datatracker.ietf.org/doc/html/rfc8032) over Curve25519 with SHA-512 and authenticates the device with a previous enrolled token instead of a password.

```
# systemd-cryptenroll /dev/disk --fido2-device=device --fido2-credential-algorithm=eddsa --unlock-fido2-device=aut
o
```

Note: Both tokens must be plugged in to the system for successful enrollment.

5.3 Trusted Platform Module

systemd-cryptenroll has native support for enrolling LUKS keys in TPMs. It requires the following:

- tpm2-tss (https://archlinux.org/packages/?name=tpm2-tss) must be installed,
- A LUKS2 device (currently the default type used by cryptsetup),
- If you intend to use this method on your root partition, some tweaks need to be made to the initramfs (see dm-crypt/System configuration#Using systemd-cryptsetup-generator for

advanced configuration) :

- mkinitcpio users: enable the `systemd` and `sd-encrypt` hooks.

  Note: The order of the entries in the hooks is important. A nonstandard ordering can make the system unbootable (https://bbs.archlinux.org/viewtopic.php?id=273945) (you will need to rebuild the initrd from within arch-chroot to recover). See dm-crypt/System configuration#Examples for an example of the correct order.

- dracut users: enable the `tpm2-tss` module.

To begin, run the following command to list your installed TPMs and the driver in use:

```
$ systemd-cryptenroll --tpm2-device=list
```

Tip: If your computer has multiple TPMs installed, specify the one you wish to use with `--tpm2-device=/path/to/tpm2_device` in the following steps.

A key may be enrolled in both the TPM and the LUKS volume using only one command. The following example generates a new random key, adds it to the volume so it can be used to unlock it in addition to the existing keys, and binds this new key to PCR 7 (Secure Boot state):

```
# systemd-cryptenroll --tpm2-device=auto /dev/sdX
```

where `/dev/sdX` is the full path to the encrypted LUKS volume. Use `--unlock-key-file=/path/to/keyfile` if the LUKS volume is unlocked by a keyfile instead of a passphrase.

Refer to systemd-cryptenroll(1) (https://man.archlinux.org/man/systemd-cryptenroll.1) and Trusted Platform Module#Accessing PCR registers for common PCR measurements in Linux. Adjust the `--tpm2-pcrs=7` default as necessary (parameters are separated by the `+` symbol).

Warning:

- Make sure Secure Boot is active and in user mode when binding to PCR 7, otherwise, unauthorized boot devices could unlock the encrypted volume.
- The state of PCR 7 can change if firmware certificates change, which can risk locking the user out. This can be implicitly done by fwupd[1] (https://github.com/systemd/systemd/blob/ed272a9ff59a26beedaab508dd3c9d631de67165/TODO#L664-L673) or explicitly by rotating Secure Boot keys.
- Only binding to PCRs measured pre-boot (PCRs 0-7) opens a vulnerability from rogue operating systems. A rogue partition with metadata copied from the real root filesystem (such as partition UUID) can mimic the original partition. Then, initramfs will attempt to mount the rogue partition as the root filesystem (decryption failure will fall back to password entry), leaving pre-boot PCRs unchanged. The rogue root filesystem with files controlled by an attacker is still able to receive the decryption key for the real root partition. See Brave New Trusted Boot World (https://0pointer.net/blog/brave-new-trusted-boot-world.html) and BitLocker documentation (https://learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker/countermeasures) for additional information.

The combination of PCRs to bind to depends on the individual case to balance usability and lock-down. For example, you may require UEFI firmware updates without manual intervention to the **Secure Boot** state, or different boot devices. As another example, Microsoft's **Bitlocker (https://learn.microsoft.com/en-us/windows-hardware/test/hlk/testref/954cf796-a640-4134-b742-eaf0ed2663ff#troubleshooting)** prefers PCR `7+11`, but may also use other PCR combinations.

> **Note:**
>
> - It is possible to require a PIN to be entered in addition to the TPM state being correct. Simply add the option `--tpm2-with-pin=yes` to the command above and enter the PIN when prompted.
> - *systemd-cryptenroll* does not check the TPM measurement before asking for the PIN, therefore consider using a unique PIN since the environment may be untrustworthy.

To check that the new key was enrolled, dump the LUKS configuration and look for a `systemd-tpm2` token entry, as well as an additional entry in the *Keyslots* section:

```
# cryptsetup luksDump /dev/sdX
```

To test that the key works, run the following command while the LUKS volume is closed:

```
# systemd-cryptsetup attach mapping_name /dev/sdX none tpm2-device=auto
```

where `mapping_name` is your chosen name for the volume once opened.

See **dm-crypt/System configuration#crypttab** and **dm-crypt/System configuration#Trusted Platform Module and FIDO2 keys** in order to unlock the volume at boot time.

> **Note:** While you may specify the UUID of your LUKS volume in place of the pathname in `/etc/crypttab`, the *systemd-cryptenroll* command itself currently only supports path names.

See **systemd-cryptenroll(1) (https://man.archlinux.org/man/systemd-cryptenroll.1)** and **crypttab(5) (https://man.archlinux.org/man/crypttab.5)** for more information and examples.

6 See also

- **Lennart's blog: Unlocking LUKS2 volumes with TPM2, FIDO2, PKCS#11 Security Hardware on systemd 248 (https://0pointer.net/blog/unlocking-luks2-volumes-with-tpm2-fido2-pkcs11-security-hardware-on-systemd-248.html)**

-