NAME

systemd-cryptenroll − Enroll PKCS#11, FIDO2, TPM2 token/devices to LUKS2 encrypted volumes

SYNOPSIS

systemd−cryptenroll [OPTIONS...] [DEVICE]

DESCRIPTION

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. Specifically, it supports tokens and credentials of the following kind to be enrolled:

1. PKCS#11 security tokens and smartcards that may carry an RSA key pair (e.g. various YubiKeys)
2. FIDO2 security tokens that implement the "hmac−secret" extension (most FIDO2 keys, including YubiKeys)
3. TPM2 security devices
4. Recovery keys. These are similar to regular passphrases, however are randomly generated on the computer and thus generally have higher entropy than user chosen passphrases. Their character set has been designed to ensure they are easy to type in, while having high entropy. They may also be scanned off screen using QR codes. Recovery keys may be used for unlocking LUKS2 volumes wherever passphrases are accepted. They are intended to be used in combination with an enrolled hardware security token, as a recovery option when the token is lost.
5. Regular passphrases

In addition, the tool may be used to enumerate currently enrolled security tokens and wipe a subset of them. The latter may be combined with the enrollment operation of a new security token, in order to update or replace enrollments.

The tool supports only LUKS2 volumes, as it stores token meta−information in the LUKS2 JSON token area, which is not available in other encryption formats.

LIMITATIONS

Note that currently when enrolling a new key of one of the five supported types listed above, it is required to first provide a passphrase or recovery key (i.e. one of the latter two key types). For example, it's currently not possible to unlock a device with a FIDO2 key in order to enroll a new FIDO2 key. Instead, in order to enroll a new FIDO2 key, it is necessary to provide an already enrolled regular passphrase or recovery key. Thus, if in future key roll−over is desired it's generally recommended to combine TPM2, FIDO2, PKCS#11 key enrollment with enrolling a regular passphrase or recovery key.

Also note that support for enrolling multiple FIDO2 tokens is currently not too useful, as while unlocking **systemd−cryptsetup** cannot identify which token is currently plugged in and thus does not know which authentication request to send to the device. This limitation does not apply to tokens enrolled via PKCS#11 — because tokens of this type may be identified immediately, before authentication.

OPTIONS

The following options are understood:

−−password
Enroll a regular password/passphrase. This command is mostly equivalent to **cryptsetup luksAddKey**, however may be combined with **−−wipe−slot=** in one call, see below.

−−recovery−key
Enroll a recovery key. Recovery keys are most identical to passphrases, but are computer generated instead of human chosen, 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.

−−pkcs11−token−uri=*URI*
Enroll a PKCS#11 security token or smartcard (e.g. a YubiKey). Expects a PKCS#11 smartcard URI referring to the token. Alternatively the special value "auto" may be specified, in order to

automatically determine the URI of a currently plugged in security token (of which there must be exactly one). The special value "list" may be used to enumerate all suitable PKCS#11 tokens currently plugged in. The security token must contain an RSA key pair which is used to encrypt the randomly generated key that is used to unlock the LUKS2 volume. The encrypted key is then stored in the LUKS2 JSON token header area.

In order to unlock a LUKS2 volume with an enrolled PKCS#11 security token, specify the **pkcs11−uri=** option in the respective /etc/crypttab line:

myvolume /dev/sda1 − pkcs11−uri=auto

See **crypttab**(5) for a more comprehensive example of a **systemd−cryptenroll** invocation and its matching /etc/crypttab line.

−−fido2−device=*PATH*
Enroll a FIDO2 security token that implements the "hmac−secret" extension (e.g. a YubiKey). Expects a hidraw device referring to the FIDO2 device (e.g. /dev/hidraw1). Alternatively the special value "auto" may be specified, in order to automatically determine the device node of a currently plugged in security token (of which there must be exactly one). The special value "list" may be used to enumerate all suitable FIDO2 tokens currently plugged in. Note that many hardware security tokens that implement FIDO2 also implement the older PKCS#11 standard. Typically FIDO2 is preferable, given it's simpler to use and more modern.

In order to unlock a LUKS2 volume with an enrolled FIDO2 security token, specify the **fido2−device=** option in the respective /etc/crypttab line:

myvolume /dev/sda1 − fido2−device=auto

See **crypttab**(5) for a more comprehensive example of a **systemd−cryptenroll** invocation and its matching /etc/crypttab line.

−−fido2−with−client−pin=*BOOL*
When enrolling a FIDO2 security token, controls whether to require the user to enter a PIN when unlocking the volume (the FIDO2 "clientPin" feature). Defaults to "yes". (Note: this setting is without effect if the security token does not support the "clientPin" feature at all, or does not allow enabling or disabling it.)

−−fido2−with−user−presence=*BOOL*
When enrolling a FIDO2 security token, controls whether to require the user to verify presence (tap the token, the FIDO2 "up" feature) when unlocking the volume. Defaults to "yes". (Note: this setting is without effect if the security token does not support the "up" feature at all, or does not allow enabling or disabling it.)

−−fido2−with−user−verification=*BOOL*
When enrolling a FIDO2 security token, controls whether to require user verification when unlocking the volume (the FIDO2 "uv" feature). Defaults to "no". (Note: this setting is without effect if the security token does not support the "uv" feature at all, or does not allow enabling or disabling it.)

−−tpm2−device=*PATH*
Enroll a TPM2 security chip. Expects a device node path referring to the TPM2 chip (e.g. /dev/tpmrm0). Alternatively the special value "auto" may be specified, in order to automatically determine the device node of a currently discovered TPM2 device (of which there must be exactly one). The special value "list" may be used to enumerate all suitable TPM2 devices currently discovered.

In order to unlock a LUKS2 volume with an enrolled TPM2 security chip, specify the **tpm2−device=** option in the respective /etc/crypttab line:

myvolume /dev/sda1 − tpm2−device=auto

See **crypttab**(5) for a more comprehensive example of a **systemd−cryptenroll** invocation and its matching /etc/crypttab line.

Use **−−tpm2−pcrs=** (see below) to configure which TPM2 PCR indexes to bind the enrollment to.

−−tpm2−pcrs= [PCR...]
Configures the TPM2 PCRs (Platform Configuration Registers) to bind the enrollment requested via **−−tpm2−device=** to. Takes a "+" separated list of numeric PCR indexes in the range 0...23. If not used, defaults to PCR 7 only. If an empty string is specified, binds the enrollment to no PCRs at all. PCRs allow binding the enrollment to specific software versions and system state, so that the enrolled unlocking key is only accessible (may be "unsealed") if specific trusted software and/or configuration is used.

Table 1. Well−known PCR Definitions

PCR	Explanation
0	Core system firmware executable code; changes on firmware updates
1	Core system firmware data/host platform configuration; typically contains serial and model numbers, changes on basic hardware/CPU/RAM replacements
2	Extended or pluggable executable code; includes option ROMs on pluggable hardware
3	Extended or pluggable firmware data; includes information about pluggable hardware
4	Boot loader; changes on boot loader updates
5	GPT/Partition table; changes when the partitions are added, modified or removed
6	Power state events; changes on system suspend/sleep
7	Secure boot state; changes when UEFI SecureBoot mode is enabled/disabled
8	**sd-boot**(7) measures the kernel command line in this PCR.

−−wipe−slot= [SLOT...]
Wipes one or more LUKS2 key slots. Takes a comma separated list of numeric slot indexes, or the special strings "all" (for wiping all key slots), "empty" (for wiping all key slots that are unlocked by an empty passphrase), "password" (for wiping all key slots that are unlocked by a traditional passphrase), "recovery" (for wiping all key slots that are unlocked by a recovery key), "pkcs11" (for wiping all key slots that are unlocked by a PKCS#11 token), "fido2" (for wiping all key slots that are unlocked by a FIDO2 token), "tpm2" (for wiping all key slots that are unlocked by a TPM2 chip), or any combination of these strings or numeric indexes, in which case all slots matching either are wiped. As safety precaution an operation that wipes all slots without exception (so that the volume cannot be unlocked at all anymore, unless the volume key is known) is refused.

This switch may be used alone, in which case only the requested wipe operation is executed. It may also be used in combination with any of the enrollment options listed above, in which case the enrollment is completed first, and only when successful the wipe operation executed — and the newly added slot is always excluded from the wiping. Combining enrollment and slot wiping may thus be used to update existing enrollments:

systemd−cryptenroll /dev/sda1 −−wipe−slot=tpm2 −−tpm2−device=auto

The above command will enroll the TPM2 chip, and then wipe all previously created TPM2 enrollments on the LUKS2 volume, leaving only the newly created one. Combining wiping and enrollment may also be used to replace enrollments of different types, for example for changing from a PKCS#11 enrollment to a FIDO2 one:

systemd−cryptenroll /dev/sda1 −−wipe−slot=pkcs11 −−fido2−device=auto

Or for replacing an enrolled empty password by TPM2:

systemd−cryptenroll /dev/sda1 −−wipe−slot=empty −−tpm2−device=auto

−h, **−−help**
Print a short help text and exit.

−−version
Print a short version string and exit.

EXIT STATUS

On success, 0 is returned, a non−zero failure code otherwise.

SEE ALSO

systemd(1), **systemd-cryptsetup@.service**(8), **crypttab**(5), **cryptsetup**(8)