systemd-creds

- Description
- Commands
- Options
- Exit status
- Examples
- See Also

systemd

systemd-creds

1

systemd-creds

Lists, shows, encrypts and decrypts service credentials

systemd-creds

OPTIONS

COMMAND

ARGS

Description

`systemd-creds` is a tool for listing, showing, encrypting and decrypting unit credentials. Credentials are limited-size binary or textual objects that may be passed to unit processes. They are primarily used for passing cryptographic keys (both public and private) or certificates, user account information or identity information from the host to services.

Credentials are configured in unit files via the `ImportCredential=`, `LoadCredential=`, `SetCredential=`, `LoadCredentialEncrypted=`, and `SetCredentialEncrypted=` settings, see `systemd.exec(5)` for details.

For further information see System and Service Credentials documentation.

Commands

The following commands are understood:

`list`
Show a list of credentials passed into the current execution context. This command shows the files in the directory referenced by the `$CREDENTIALS_DIRECTORY` environment variable, and is intended to be executed from within service context.

Along with each credential name, the size and security state is shown. The latter is one of `secure` (in case the credential is backed by unswappable memory, i.e. `ramfs`), `weak` (in case it is backed by any other type of memory), or `insecure` (if having any access mode that is not 0400, i.e. if readable by anyone but the owner).

`cat` <credential...>
Show contents of specified credentials passed into the current execution context. Takes one or more credential names, whose contents shall be written to standard output.

When combined with `--json=` or `--transcode=` the output is transcoded in simple ways before outputting.

`setup`
Generates a host encryption key for credentials, if one has not been generated already. This ensures the `/var/lib/systemd/credential.secret` file is initialized with a random secret key if it does not exist yet. This secret key is used when encrypting/decrypting credentials with `encrypt` or `decrypt`, and is only accessible to the root user. Note that there's typically no need to invoke this command explicitly as it is implicitly called when `encrypt` is invoked, and credential host key encryption selected.

`encrypt` <input|-> <output|->
Loads the specified (unencrypted plaintext) input credential file, encrypts it and writes the (encrypted ciphertext) output to the specified target credential file. The resulting file may be referenced in the `LoadCredentialEncrypted=` setting in unit files, or its contents used literally in `SetCredentialEncrypted=` settings.

Takes two file system paths. The file name part of the output path is embedded as name in the encrypted credential, to ensure encrypted credentials cannot be renamed and reused for different purposes without this being noticed. The credential name to embed may be overridden with the `--name=` setting. The input or output paths may be specified as `-`, in which case the credential data is read from/written to standard input and standard output. If the output path is specified as `-` the credential name cannot be derived from the file system path, and thus should be specified

explicitly via the `--name=` switch.

The credential data is encrypted and authenticated symmetrically with one of the following encryption keys:

1. A secret key automatically derived from the system's TPM2 chip. This encryption key is not stored on the host system and thus decryption is only possible with access to the original TPM2 chip. Or in other words, the credential secured in this way can only be decrypted again by the local machine.
2. A secret key stored in the `/var/lib/systemd/credential.secret` file which is only accessible to the root user. This "host" encryption key is stored on the host file system, and thus decryption is possible with access to the host file system and sufficient privileges. The key is automatically generated when needed, but can also be created explicitly with the `setup` command, see above.
3. A combination of the above: an encryption key derived from both the TPM2 chip and the host file system. This means decryption requires both access to the original TPM2 chip and the OS installation. This is the default mode of operation if a TPM2 chip is available and `/var/lib/systemd/` resides on persistent media.

Which of the three keys shall be used for encryption may be configured with the `--with-key=` switch. Depending on the use-case for the encrypted credential the key to use may differ. For example, for credentials that shall be accessible from the initrd, encryption with the host key is not appropriate, since access to the host key is typically not available from the initrd. Thus, for such credentials only the TPM2 key should be used.

Encrypted credentials are always encoded in Base64.

Use `decrypt` (see below) to undo the encryption operation, and acquire the decrypted plaintext credential from the encrypted ciphertext credential.

The credential data is encrypted using AES256-GCM, i.e. providing both confidentiality and integrity, keyed by a SHA256 hash of one or both of the secret keys described above.

`decrypt` <input|-> [<output|->]
Undoes the effect of the `encrypt` operation: loads the specified (encrypted ciphertext) input credential file, decrypts and authenticates it and writes the (decrypted plaintext) output to the specified target credential file.

Takes one or two file system paths. The file name part of the input path is compared with the credential name embedded in the encrypted file. If it does not match decryption fails. This is done in order to ensure that encrypted credentials are not re-purposed without this being detected. The credential name to compare with the embedded credential name may also be overridden with the `--name=` switch. If the input path is specified as `-`, the encrypted credential is read from standard input. If only one path is specified or the output path specified as `-`, the decrypted credential is written to standard output. In this mode, the expected name embedded in the credential cannot be derived from the path and should be specified explicitly with `--name=`.

Decrypting credentials requires access to the original TPM2 chip and/or credentials host key, see above. Information about which keys are required is embedded in the encrypted credential data, and thus decryption is entirely automatic.

Options

`--system`
When specified with the `list` and `cat` commands operates on the credentials passed to system as a whole instead of on those passed to the current execution context. This is useful in container environments where credentials may be passed in from the container manager.

`--user`
When specified with the `encrypt` and `decrypt` commands encrypts a user-scoped (rather than a system-scoped) credential. Use `--uid=` to select which user the credential is from. Such credentials may only be decrypted from the specified user's context, except if privileges can be acquired. Generally, when an encrypted credential shall be used in the per-user service manager it should be encrypted with this option set, when it shall be used in the system service manager it should be encrypted without.

Internally, this ensures that the selected user's numeric UID and username, as well as the system's `machine-id(5)` are incorporated into the encryption key.

`--uid=`
Specifies the user to encrypt the credential for. Takes a user name or numeric UID. If set, implies `--user`. If set to the special string `self` sets the user to the user of the calling process. If `--user` is used without `--uid=` then `--uid=self` is implied, i.e. the credential is encrypted for the calling user.

`--transcode=`
When specified with the `cat` or `decrypt` commands, transcodes the output before showing it. Takes one of `base64`, `unbase64`, `hex` or `unhex` as argument, in order to encode/decode the credential data with Base64 or as series of hexadecimal values. The special value `help` may be used to list supported transcode types.

Note that this has no effect on the `encrypt` command, as encrypted credentials are unconditionally encoded in Base64.

`--newline=`
When specified with `cat` or `decrypt` controls whether to add a trailing newline character to the end of the output if it does not end in one, anyway. Takes one of `auto`, `yes` or `no`. The default mode of `auto` will suffix the output with a single newline character only when writing credential data to a TTY.

`--pretty`; `-p`
When specified with `encrypt` controls whether to show the encrypted credential as

`SetCredentialEncrypted=` setting that may be pasted directly into a unit file. Has effect only when used together with `--name=` and `-` as the output file.

`--name=name`
When specified with the `encrypt` command controls the credential name to embed in the encrypted credential data. If not specified, the name is chosen automatically from the filename component of the specified output path. If specified as empty string no credential name is embedded in the encrypted credential, and no verification of credential name is done when the credential is decrypted.

When specified with the `decrypt` command control the credential name to validate the credential name embedded in the encrypted credential with. If not specified, the name is chosen automatically from the filename component of the specified input path. If no credential name is embedded in the encrypted credential file (i.e. the `--name=` with an empty string was used when encrypted) the specified name has no effect as no credential name validation is done.

Embedding the credential name in the encrypted credential is done in order to protect against reuse of credentials for purposes they were not originally intended for, under the assumption the credential name is chosen carefully to encode its intended purpose.

`--timestamp=timestamp`
When specified with the `encrypt` command controls the timestamp to embed into the encrypted credential. Defaults to the current time. Takes a timestamp specification in the format described in `systemd.time(7)`.

When specified with the `decrypt` command controls the timestamp to use to validate the "not-after" timestamp that was configured with `--not-after=` during encryption. If not specified, defaults to the current system time.

`--not-after=timestamp`
When specified with the `encrypt` command controls the time when the credential shall not be used anymore. This embeds the specified timestamp in the encrypted credential. During decryption the timestamp is checked against the current system clock, and if the timestamp is in the past the decryption will fail. By default, no such timestamp is set. Takes a timestamp specification in the format described in `systemd.time(7)`.

`--with-key=`; `-H`; `-T`
When specified with the `encrypt` command controls the encryption/signature key to use. Takes one of `host`, `tpm2`, `host+tpm2`, `null`, `auto`, `auto-initrd`. See above for details on the key types. If set to `auto` (which is the default) the TPM2 key is used if a TPM2 device is found and not running in a container. The host key is used if `/var/lib/systemd/` is on persistent media. This means on typical systems the encryption is by default bound to both the TPM2 chip and the OS installation, and both need to be available to decrypt the credential again. If `auto` is selected but neither TPM2 is available (or running in container) nor `/var/lib/systemd/` is on persistent media, encryption will fail. If set to `null` a fixed zero length key is used (thus, in this mode no confidentiality nor authenticity are provided!). This logic is useful to cover for systems that lack a TPM2 chip but where credentials

shall be generated. Note that decryption of such credentials is refused on systems that have a TPM2 chip and where UEFI SecureBoot is enabled (this is done so that such a locked down system cannot be tricked into loading a credential generated this way that lacks authentication information. If either UEFI SecureBoot or a TPM2 are not available, then loading such credentials is allowed by default). If set to `auto-initrd` a TPM2 key is used if a TPM2 is found. If not, a fixed zero length key is used, equivalent to `null` mode. This option is particularly useful to generate credentials files that are encrypted/authenticated against TPM2 where available but still work on systems lacking support for this. The special value `help` may be used to list supported key types.

systemd v262 and later additionally pin the encrypted credential to the TPM2's SRK to defend against MITM interposer attacks when decrypting credentials. This makes the resulting credentials incompatible with earlier systemd versions.

The `-H` switch is a shortcut for `--with-key=host`. Similar, `-T` is a shortcut for `--with-key=tpm2`.

When encrypting credentials that shall be used in the initrd (where `/var/lib/systemd/` is typically not available) make sure to use `--with-key=auto-initrd` mode, to disable binding against the host secret.

This switch has no effect on the `decrypt` command, as information on which key to use for decryption is included in the encrypted credential already.

`--tpm2-device=PATH`
Controls the TPM2 device to use. 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 suitable TPM2 device (of which there must be exactly one). The special value `list` may be used to enumerate all suitable TPM2 devices currently discovered.

`--tpm2-pcrs=PCR+PCR...`
Configures the TPM2 PCRs (Platform Configuration Registers) to bind the encryption key to. Takes a `+` separated list of numeric PCR indexes in the range 0…23. If an empty string is specified, binds the encryption key to no PCRs at all (this is also the default if this option is not used). For details about the PCRs available, see the documentation of the switch of the same name for `systemd-cryptenroll(1)`.

`--tpm2-public-key=PATH`; `--tpm2-public-key-pcrs=PCR+PCR...`
Configures a TPM2 signed PCR policy to bind encryption to, for use with the `encrypt` command. The `--tpm2-public-key=` option accepts a path to a PEM encoded RSA public key, to bind the encryption to. If this is not specified explicitly, but a file `tpm2-pcr-public-key.pem` exists in one of the directories `/etc/systemd/`, `/run/systemd/`, `/usr/lib/systemd/` (searched in this order), it is automatically used. The `--tpm2-public-key-pcrs=` option takes a list of TPM2 PCR indexes to bind to (same syntax as `--tpm2-pcrs=` described above). If not specified, defaults to 11 (i.e. this binds the policy to any unified kernel image for which a PCR signature can be provided).

Note the difference between `--tpm2-pcrs=` and `--tpm2-public-key-pcrs=`: the former binds decryption to the current, specific PCR values; the latter binds decryption to any set of PCR values for which a signature by the specified public key can be provided. The latter is hence more useful

in scenarios where software updates shall be possible without losing access to all previously encrypted secrets.

`--tpm2-signature=PATH`
Takes a path to a TPM2 PCR signature file as generated by the `systemd-measure(1)` tool and that may be used to allow the `decrypt` command to decrypt credentials that are bound to specific signed PCR values. If this is not specified explicitly, and a credential with a signed PCR policy is attempted to be decrypted, a suitable signature file `tpm2-pcr-signature.json` is searched for in `/etc/systemd/`, `/run/systemd/`, `/usr/lib/systemd/` (in this order) and used.

`--allow-null`
Allow decrypting credentials that use a null key. By default decryption of credentials encrypted/authenticated with a null key is only allowed if UEFI SecureBoot is off or if a TPM2 is not available.

`--refuse-null`
Refuse decrypting credentials that use a null key, regardless of the UEFI SecureBoot state or TPM2 availability (see above).

`--quiet`; `-q`
Suppress additional output.

Exit status

On success, 0 is returned.

Examples

Encrypt a password for use as credential

The following command line encrypts the specified password `hunter2`, writing the result to a file `password.cred`.

```
# echo -n hunter2 | systemd-creds encrypt - password.cred
```

This decrypts the file `password.cred` again, revealing the literal password:

```
# systemd-creds decrypt password.cred
hunter2
```

Encrypt a password and include it in a unit file

The following command line prompts the user for a password and generates a `SetCredentialEncrypted=` line from it for a credential named `mysql-password`, suitable for inclusion in a unit file.

```
# systemd-ask-password -n | systemd-creds encrypt --name=mysql-password -p - -
□□Password: ****
SetCredentialEncrypted=mysql-password: \
        k6iUCUh0RJCQyvL8k8q1UyAAAAABAAAADAAAABAAAAASfFsBoPLIm/dlDoGAAAAAAAAAA \
        NAAAAAgAAAAAH4AILIOZ3w6rTzYsBy9G7liaCAd4i+Kpvs8mAgArzwuKxd0ABDjgSeO5k \
        mKQc58zM94ZffyRmuNeX1lVHE+9e2YD87KfRFNoDLS7F3YmCb347gCiSk2an9egZ7Y0Xs \
        700Kr6heqQswQEemNEc62k9RJnEl2q7SbcEYguegnPQUATgAIAAsAAAASACA/B90W7E+6 \
        yAR9NgiIJvxr9bpElztwzB5lUJAxtMBHIgAQACCaSV9DradOZz4EvO/LSaRyRSq2Hj0ym \
        gVJk/dVzE8Uxj8H3RbsT7rIBH02CIgm/Gv1ukSXO3DMHmVQkDG0wEciyageTfrVEer8z5 \
        9cUQfM5ynSaV2UjeUWEHuz4fwDsXGLB9eELXLztzUU9nsAyLvs3ZRR+eEK/A==
```

The generated line can be pasted 1:1 into a unit file, and will ensure the acquired password will be made available in the `$CREDENTIALS_DIRECTORY``/mysql-password` credential file for the started service.

Utilizing the unit file drop-in logic this can be used to securely pass a password credential to a unit. A similar, more comprehensive set of commands to insert a password into a service `xyz.service`:

```
# mkdir -p /etc/systemd/system/xyz.service.d
# systemd-ask-password -n | ( echo "[Service]" && systemd-creds encrypt --name=mysql-password
-p - - ) >/etc/systemd/system/xyz.service.d/50-password.conf
# systemctl daemon-reload
# systemctl restart xyz.service
```

See Also

`systemd(1)`, `systemd.exec(5)`, `systemd-measure(1)`

□□□ #1
□ □□□ □□□ 2026-10-31 14:59:34 UTC
□ □□□ □□□ 2026-10-31 14:59:34 UTC