Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'clevis.1' command

$ man clevis.1

CLEVIS(1) CLEVIS(1)

NAME

clevis - Automated decryption policy framework

SYNOPSIS

clevis COMMAND [OPTIONS]

OVERVIEW

Clevis is a framework for automated decryption policy. It allows you to define a policy at encryption time that must be satisfied for the data to decrypt. Once this policy is met, the data is decrypted.

Clevis is pluggable. Our plugins are called pins. The job of a pin is to take a policy as its first argument and plaintext on standard input and to encrypt the data so that it can be automatically decrypted if the policy is met. Lets walk through an example.

TANG BINDING

Clevis provides support for the Tang network binding server. Tang provides a stateless, lightweight alternative to escrows. Encrypting data using the Tang pin works much like our HTTP pin above:

$ clevis encrypt tang '{"url":"http://tang.srv"}' < PT > JWE

The advertisement contains the following signing keys:

_OsIk0T-E2l6qjfdDiwVmidoZjA

Do you wish to trust these keys? [ynYN] y

As you can see above, Tang utilizes a trust-on-first-use workflow.

Alternatively, Tang can perform entirely offline encryption if you

pre-share the server advertisement. Decryption, too works like our first example:

```
$ clevis decrypt < JWE > PT
```

For more information, see clevis-encrypt-tang(1).

TPM2 BINDING

Clevis provides support to encrypt a key in a Trusted Platform Module 2.0 (TPM2) chip. The cryptographically-strong, random key used for encryption is encrypted using the TPM2 chip, and then at decryption time is decrypted using the TPM2 to allow clevis to decrypt the secret stored in the JWE.

Encrypting data using the tpm2 pin works the same than the pins mentioned above:

```
$ clevis encrypt tpm2 '{}' < PT > JWE
```

The pin has reasonable defaults for its configuration, but a different hierarchy, hash, and key algorithms can be chosen if the defaults used are not suitable.

Decryption also works similar to other pins, only the JWE needs to be provided:

```
$ clevis decrypt < JWE > PT
```

Note that like other pins no configuration is used for decryption, this is due clevis storing the public and private keys to unseal the TPM2 encrypted object in the JWE so clevis can fetch that information from there.

For more information see clevis-encrypt-tpm2(1).

SHAMIR?S SECRET SHARING

Clevis provides a way to mix pins together to create sophisticated unlocking and high availability policies. This is accomplished by using an algorithm called Shamir?s Secret Sharing (SSS).

SSS is a thresholding scheme. It creates a key and divides it into a number of pieces. Each piece is encrypted using another pin (possibly even SSS recursively). Additionally, you define the threshold t. If at least t pieces can be decrypted, then the encryption key can be recovered and decryption can succeed.

For example, let?s create a high-availability setup using Tang:

```
$ cfg='{"t":1,"pins":{"tang":[{"url":...},{"url":...}]}}'
$ clevis encrypt sss "$cfg" < PT > JWE
```

In this policy, we are declaring that we have a threshold of 1, but that there are multiple key fragments encrypted using different Tang servers. Since our threshold is 1, so long as any of the Tang servers are available, decryption will succeed. As always, decryption is simply:

```
$ clevis decrypt < JWE > PT
```

For more information, see clevis-encrypt-tang(1).

LUKS BINDING

Clevis can be used to bind an existing LUKS volume to its automation policy. This is accomplished with a simple command:

```
$ clevis luks bind -d /dev/sda tang '{"url":...}'
```

This command performs four steps:

1. Creates a new key with the same entropy as the LUKS master key ? maximum entropy bits is 256.
2. Encrypts the new key with Clevis.
3. Stores the Clevis JWE in the LUKS header.
4. Enables the new key for use with LUKS.

This disk can now be unlocked with your existing password as well as with the Clevis policy. Clevis provides two unlockers for LUKS volumes. First, we provide integration with Dracut to automatically unlock your root volume during early boot. Second, we provide integration with UDisks2 to automatically unlock your removable media in your desktop session.

For more information, see clevis-luks-bind(1).

SEE ALSO

clevis-encrypt-tang(1), clevis-encrypt-tpm2(1), clevis-encrypt-sss(1), clevis-luks-bind(1), clevis-decrypt(1)