Oracle Linux

Enabling Network-Bound Disk Encryption

F43691-10
June 2025

Oracle Linux Enabling Network-Bound Disk Encryption,

F43691-10

Contents

Preface

Oracle Linux: Enabling Network-Bound Disk Encryption provides information about the Tang server and Clevis framework, used to implement Network-Bound Disk Encryption (NBDE) on Oracle Linux systems. The instructions provided in this document are tested on Oracle Linux 8 and on Oracle Linux 9 but the tools and services are equally available for Oracle Linux 7 and work similarly. Commands to install software using `dnf` can be substituted with `yum` equivalents where required.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

About Network-Bound Disk Encryption

Network-Bound Disk Encryption (NBDE) is a method that's used to automatically decrypt Linux Unified Key Setup (LUKS) encrypted disks or volumes by using a network-based resource to obtain the information that's required to perform decryption. This facility extends LUKS, which is commonly used to encrypt disks and volumes on Oracle Linux to provide an extra mode of security, and provide automated decryption at boot.

LUKS is useful for encrypting data and protecting it when it's not in use. For server systems, LUKS can pose a problem if the root partition is encrypted because it requires a passphrase to be entered at boot to decrypt the data. NBDE helps to solve this issue by using a network-based resource to obtain a key that LUKS uses to perform the decryption automatically at boot. In this security model, the data on the disk is protected if the system boots within a trusted network. If the disk is removed from the server and isn't booted within the trusted network, the data remains encrypted, unless the usual LUKS passphrase is provided.

NBDE is achieved on Oracle Linux by configuring a Tang server on a trusted network and then installing the Clevis decryption software on the client host where the encryption is used. Tang runs a web-based service that uses HTTP to advertise a public signing key that's used by clients to generate key pairs which are used to encrypt data. The Clevis client generates a strong cryptographic key pair, using the signing key that's provided by the Tang server, to perform an encryption. Encryption is performed by using the generated private key, which is discarded after encryption is complete, thereby protecting the data until the private key is reconstituted.

The Clevis client uses an ephemeral key to obtain the information that's required from the Tang server to reconstitute the private key so that it can decrypt the data. This process is known as the McCallum-Relyea exchange and has the advantage of avoiding key escrow, which can result in management overhead and can also introduce security risks. In this key exchange process, the keys that are used to encrypt the data on the client side are never directly shared with the Tang server and never move across the network. All the information that's exchanged is either public or encrypted by nature, which means that no TLS is required either.

Because LUKS can store several keys in different slots that are used to decrypt data, the primary passphrase that's used to lock a disk or a volume can be maintained alongside the key that's provided for NBDE. At boot, the usual passphrase prompt is displayed for LUKS decryption so that you can enter a passphrase, but if Clevis can contact the Tang server no user input is required and the passphrase prompt closes automatically after the key is decrypted and access is granted to the data that's stored on the disk or volume.

More information about the Tang server is available at https://github.com/latchset/tang.

More information about the Clevis framework is available at https://github.com/latchset/clevis.

General information about public key cryptography is available in Oracle Linux: Managing Certificates and Public Key Infrastructure.

Information about using LUKS for disk encryption is available in Oracle Linux 10: Managing Storage Devices, Oracle Linux 9: Managing Storage Devices, and Oracle Linux 8: Managing Storage Devices

2

Install and Configure a Tang Server

A Tang server is used to store and manage encryption keys that are accessed by a Clevis client to perform automatic encryption and decryption. The following instructions describe how to install and configure the Tang server and how to initialize it with a set of signing keys.

Install the Tang Package and Enable the Tang Socket in Systemd

1. Install the Tang package and related dependencies.

   ```
   sudo dnf install -y tang
   ```

2. Enable the Tang socket in Systemd.

   ```
   sudo systemctl enable tangd.socket
   ```

To configure Tang to run on a TCP port other than the default port, you must change the Systemd configuration. You can change the TCP port by following the instructions in Optionally Configure the Tang Server to Run on a Specified Port. Otherwise, see Update the Firewall Policy to allow network traffic to reach the Tang server.

Optionally Configure the Tang Server to Run on a Specified Port

What Do You Need?

- The Tang package must be installed, and the Tang socket must be enabled in Systemd. See Install the Tang Package and Enable the Tang Socket in Systemd for information on how to do this.
- In Oracle Linux 10, you need to install the `policycoreutils-python-utils` package before you can use the SELinux `semanage` command. You can install the package by running the following command:

  ```
  sudo dnf install policycoreutils-python-utils
  ```

By default, Tang runs on TCP port 80. If you intend to override this setting to run a Tang server on another port, you must configure SELinux so that Tang works with the required TCP port.

1. Configure Tang to listen on a chosen TCP port.

   For example, to configure Tang to listen on TCP port 7500, run:

   ```
   sudo semanage port -a -t tangd_port_t -p tcp 7500
   ```

2. Configure a Systemd override so that Tang listens on the chosen port.

If you have chosen to run Tang on a different port, you must configure a Systemd override by configuring a Socket entry:

```
sudo mkdir -p /etc/systemd/system/tangd.socket.d/
sudo echo | sudo tee /etc/systemd/system/tangd.socket.d/port.conf  >/dev/
null <<EOF
[Socket]
ListenStream=
ListenStream=7500
EOF
```

3. Reload Systemd daemon configuration.

   If you edit Systemd configuration you must reload Systemd daemon configuration for the changes to take effect:

   ```
   sudo systemctl daemon-reload
   ```

You must configure the firewall to allow traffic to reach the configured TCP port. See Update the Firewall Policy to configure the firewall.

Update the Firewall Policy

1. Set the firewall policy appropriate for the TCP port that's used by Tang.
   - If you're using the default port, you can run:

     ```
     sudo firewall-cmd --add-service=http
     ```

   - If you aren't using the default port for the Tang server and have configured the server to run on a specified port number, such as port 7500, run:

     ```
     sudo firewall-cmd --add-port=7500/tcp
     ```

2. Make the firewall policy permanent.

   ```
   sudo firewall-cmd --runtime-to-permanent
   ```

If you haven't already started the Systemd socket, do this so that the Tang service can run on the configured port. See Start and Enable the Systemd Tang Socket for more information.

Start and Enable the Systemd Tang Socket

1. Start and enable the Tang socket so that incoming connections can access the Tang server.

   ```
   sudo systemctl enable --now tangd.socket
   ```

   The Tang process runs when a client connects to the configured socket.

2. Verify that the Tang socket is started and enabled.

   ```
   sudo systemctl status tangd.socket
   ```

The output must display that the socket is active and listening.

If you haven't initialized Tang signing keys, do this now so that the Tang service can advertise these signing keys on the network. See Initialize Tang Signing Keys for more information.

Initialize Tang Signing Keys

1. Initialize keys for Tang to use.

   ```
   sudo /usr/libexec/tangd-keygen /var/db/tang
   ```

2. Verify that the keys are advertised on the Tang server.

   Check that one of the keys in `/var/db/tang` is advertised by the Tang server on the port for which you have configured it to listen, for example:

   ```
   sudo tang-show-keys 7500
   ```

   You can use the output from this command to validate the key hash that's used when you configure a client to use this Tang server.

For better security, rotate the Tang keys periodically so that they don't become stale. See Rotate Tang Keys for more information.

If the Tang server is configured and running, you can install Clevis on client systems and configure automatic decryption for any LUKS encrypted devices on these host systems. See Perform Automated Encryption and Decryption With Clevis for more information.

Rotate Tang Keys

Rotate Tang keys periodically to improve security. Rotation of keys is manual and isn't a mandatory procedure.

1. Rename old keys in `/var/db/tang`.

   To rotate Tang keys, rename old keys within the `/var/db/tang` directory to prefix them with a period (.), so that they're hidden, and rerun the initialization command. For example:

   ```
   sudo bash -c 'cd /var/db/tang; for i in *; do mv $i .$i; done'
   ```

2. Regenerate the Tang keys.

   ```
   sudo /usr/libexec/tangd-keygen /var/db/tang
   ```

3. Remove old keys from the system.

   When you're certain that no client systems are still dependent on any of the old keys you can remove them from the system. Removing the old keys while clients are still using them results in failures to unlock a disk or volume and the user must provide an existing LUKS passphrase manually at boot.

   To remove the old keys, remove the files in `/var/db/tang` that are prefixed with a period (.)

When you rotate keys on the Tang server, you must update clients to use the new keys. See Update Clevis for Tang Key Rotation for more information.

3

Perform Automated Encryption and Decryption With Clevis

Clevis is client software that can perform automated decryption by using different plugin provider services. Clevis works with the Tang server provider and can handle encryption and decryption operations securely while avoiding key escrow.

You can use Clevis with LUKS to automatically unlock encrypted storage. Tools are also provided to integrate with Dracut so that you can update the initrd boot image to enable Clevis to be used at boot to automatically decrypt a device, if the system has access to the Tang server.

Install and Test Clevis

The following instructions describe how to install the Clevis client software on an Oracle Linux instance, how to test the Clevis software by performing a basic encryption task using keys provided by a Tang server, and how to update existing initrd boot images to integrate with the Clevis client for automatic decryption of LUKS encrypted partitions at boot time.

1. Install the `clevis` package and related dependencies.

   ```
   sudo dnf install -y clevis clevis-luks clevis-udisks2 clevis-dracut
   ```

   Each package has a different function:

   - `clevis` provides the basic decryption client that can communicate with a Tang server
   - `clevis-luks` is required to integrate Clevis with LUKS to perform automatic disk or volume decryption
   - `clevis-udisks2` is required to integrate Clevis with the udisks framework that's used for removable storage so the Clevis can trigger to perform LUKS decryption on removable disks
   - `clevis-dracut` is required to integrate Clevis with Dracut so that the Clevis tools can be included in initrd images for early boot integration

2. Test Clevis.

   To test that Clevis can encrypt data using the keys provided by the Tang server, follow these steps:

   a. Create a plain text file with some content that you intend to encrypt.

      ```
      echo "this is my secret message" > unencrypted.txt
      ```

   b. Encrypt the plain text file using the Tang server that you have set up and configured on the trusted network and pipe the output into an a separate file. Note that the command also prompts you to trust the signing key.

      ```
      clevis encrypt tang '{"url":"http://tang-server.example.org:7500"}' <
      unencrypted.txt > secret.jwe
      ```

```
...
The advertisement contains the following signing keys:

i9sPMu_sn6vMjzyJm8ZALj7opDE

Do you wish to trust these keys? [ynYN] Y
```

The Tang server responds to the request by providing a signing key that's used to restore the strong cryptographic key that's used to encrypt the data during the provisioning process.

c. Inspect the encrypted content that's returned by the `clevis encrypt` command.

```
cat secret.jwe
eyJhbGciOiJFQ0RILUVTIiwiY2xldmlzIjp7InBpbiI6InRhbmciLCJ0YW5n
Ijp7ImFkdiI6eyJrZXlzIjpbeyJhbGciOiJFQ01SIiwiY3J2IjoiUC01MjEi
LCJrZXlfb3BzIjpbImRlcml2ZUtleSJdLCJrdHkiOiJFQyIsIngiOiJBUVNs
...
bTJIWkpva19ETXZXTzVBejN0Zzg0dzBRd01xam9pczVnVFNzZlhTbERyNUVI
```

d. Decrypt the data in the encrypted file to ensure that decryption is possible.

```
cat secret.jwe |clevis decrypt
this is my secret message
```

The `clevis decrypt` command uses metadata that's stored during the encryption process to derive the key used to decrypt the data through information returned from a POST request to the Tang server.

3. Update initrd boot images for Clevis integration.

The `clevis-dracut` package is installed so that Clevis operations can be introduced into the initrd boot image for early boot decryption operations. After you have installed this package, you can run the `dracut` command to rebuild an existing initrd boot image to integrate it with Clevis:

```
sudo dracut -f
```

This step makes it possible for Clevis to unlock a LUKS encrypted partition at boot time, if the Tang server is accessible. Note that you're still prompted for a LUKS passphrase at boot; but, if one isn't provided and Clevis can contact the Tang server, LUKS can unlock the device and the passphrase prompt closes after a period.

Use Clevis With LUKS

You can bind Clevis to a LUKS slot for any volume or device that's LUKS encrypted. When Clevis is bound to a LUKS slot, automatic network-bound decryption is triggered when a user is prompted for a LUKS passphrase entry.

The following instructions explain how to bind and unbind Clevis against a LUKS slot, verify that Clevis is integrated with LUKS for a volume or device and update Clevis for a volume or device if the Tang keys are rotated.

Bind Clevis to a LUKS Slot

To bind Clevis to a LUKS slot to unlock a LUKS encrypted device by using a Tang server, run the `clevis luks bind` command. Note that the command prompts you to trust the key that's being advertised by the Tang server. Likewise, the command prompts you for the LUKS password.

When typing the command, provide the path to the device that's LUKS encrypted. In the following example, the system uses LVM and the root volume is LUKS encrypted, so */dev/ol/root* is used as the device path. You could equally use a block device such as `/dev/sda1`. Also, you must provide the URL to the Tang server in a JSON string.

See the following example:

```
sudo clevis luks bind -d /dev/ol/root tang '{"url": "http://tang-
server.example.org:7500"}'
...
The advertisement is signed with the following keys:
        i9sPMu_sn6vMjzyJm8ZALj7opDE

Do you wish to trust the advertisement? [yN] y
Enter existing LUKS password:
```

This operation performs several steps:

- Clevis creates a new key with the same entropy as the primary LUKS key.
- The new key is encrypted by Clevis using the Tang key.
- Clevis stores the token and metadata to contact the Tang server in the LUKS header.
- The key is enabled for use with LUKS.

Verify Clevis Integration With LUKS

You can verify Clevis integration with LUKS in several ways.

1. Check the LUKSMeta information for a device by using the `cryptsetup` command.

   ```
   sudo cryptsetup luksDump /dev/ol/root
   ```

   The output displays that the Clevis key has been added to one of the slots and a `clevis` token is assigned for that slot.

2. Check which slot is used by Clevis and the Tang server information for the binding by running the `clevis luks list` command.

   ```
   sudo clevis luks list -d /dev/ol/root
   ```

3. Check whether the device is automatically unlocked at boot.

   If the device is required at boot and `clevis-dracut` is installed and configured, the system continues to prompt you for a LUKS passphrase at boot; however, Clevis tries to decrypt its key by using the Tang server and unlocks the device automatically. The passphrase prompt closes after a period if Clevis is successful.

Update Clevis for Tang Key Rotation

Periodically a Tang server administrator might rotate Tang keys on the server for more security. When this happens, it's possible that Clevis is unable to decrypt the LUKS token correctly and this must be regenerated using the new Tang key. If Clevis fails to decrypt the LUKS token, you must authenticate to LUKS using passphrase entry at boot.

To check whether Tang keys have been rotated and to regenerate a newly encrypted token, perform the following steps:

1. Identify the LUKS slot that Clevis is bound to for a specified device.

   ```
   sudo clevis luks list -d /dev/ol/root
   ```

   Output similar to the following might be displayed:

   ```
   1: tang '{"url":"http://tang-server.example.org:7500"}'
   ```

2. Get Clevis to report whether the Tang key has been updated on the specified slot and automatically regenerate the LUKS token.

   ```
   sudo clevis luks report -d /dev/ol/root -s 1
   ```

   Output similar to the following might be displayed:

   ```
   ...
   Report detected that some keys were rotated.
   Do you want to regenerate luks metadata with
   "clevis luks regen -d /dev/ol/root -s 1"? [ynYN]
   ```

   Enter `y` to tell Clevis to automatically regenerate the LUKS token.

Note that you can optionally unbind Clevis from an existing LUKS slot and then bind it again if the instructions to regenerate a token don't work for you.

Unbind Clevis From a LUKS Slot

You can unbind Clevis from a LUKS slot by using the `clevis luks unbind` command:

Note that you must specify the slot number that Clevis is bound to by specifying the `-s` option, followed by the slot number. Be aware that this operation is destructive and wipes the LUKSMeta data for the slot that's specified, so you're prompted to confirm the operation.

1. Use the `clevis luks unbind` command to remove Clevis LUKS metadata from a slot.

   ```
   sudo clevis luks unbind -d /dev/ol/root -s 1
   ```

2. Verify that the slot is clear of metadata in LUKS.

   ```
   sudo cryptsetup luksDump /dev/ol/root
   ```

   The output displays that no metadata exists for clevis in the slot.