Red Hat Enterprise Linux 9.4 Beta

Security hardening

Enhancing security of Red Hat Enterprise Linux 9 systems

Last Updated: 2024-03-27

Red Hat Enterprise Linux 9.4 Beta Security hardening

Enhancing security of Red Hat Enterprise Linux 9 systems

Legal Notice

Abstract

Learn the processes and practices for securing Red Hat Enterprise Linux servers and workstations against local and remote intrusion, exploitation, and malicious activity. By using these approaches and tools, you can create a more secure computing environment for the data center, workplace, and home.

Table of Contents

RHEL BETA RELEASE

Red Hat provides Red Hat Enterprise Linux Beta access to all subscribed Red Hat accounts. The purpose of Beta access is to:

- Provide an opportunity to customers to test major features and capabilities prior to the general availability release and provide feedback or report issues.
- Provide Beta product documentation as a preview. Beta product documentation is under development and is subject to substantial change.

Note that Red Hat does not support the usage of RHEL Beta releases in production use cases. For more information, see What does Beta mean in Red Hat Enterprise Linux and can I upgrade a RHEL Beta installation to a General Availability (GA) release?.

BETA CHANGES

The following list highlights parts of the documentation that have been newly added or updated as part of this Beta preview:

- Configuring Keylime registrar as a container and Configuring Keylime verifier as a container describe how to install and configure the two Keylime server components, the registrar and verifier, as containers.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the Jira website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

CHAPTER 1. ENSURING SYSTEM INTEGRITY WITH KEYLIME

With Keylime, you can continuously monitor the integrity of remote systems and verify the state of systems at boot. You can also send encrypted files to the monitored systems, and specify automated actions triggered whenever a monitored system fails the integrity test.

1.1. HOW KEYLIME WORKS

You can deploy Keylime agents to perform one or more of the following actions:

Runtime integrity monitoring

Keylime runtime integrity monitoring continuously monitors the system on which the agent is deployed and measures the integrity of the files included in the allowlist and not included in the excludelist.

Measured boot

Keylime measured boot verifies the system state at boot.

Keylime's concept of trust is based on the Trusted Platform Module (TPM) technology. A TPM is a hardware, firmware, or virtual component with integrated cryptographic keys. By polling TPM quotes and comparing the hashes of objects, Keylime provides initial and runtime monitoring of remote systems.

IMPORTANT

Keylime running in a virtual machine or using a virtual TPM depends upon the integrity of the underlying host. Ensure you trust the host environment before relying upon Keylime measurements in a virtual environment.

Keylime consists of three main components:

Verifier

Initially and continuously verifies the integrity of the systems that run the agent.

Registrar

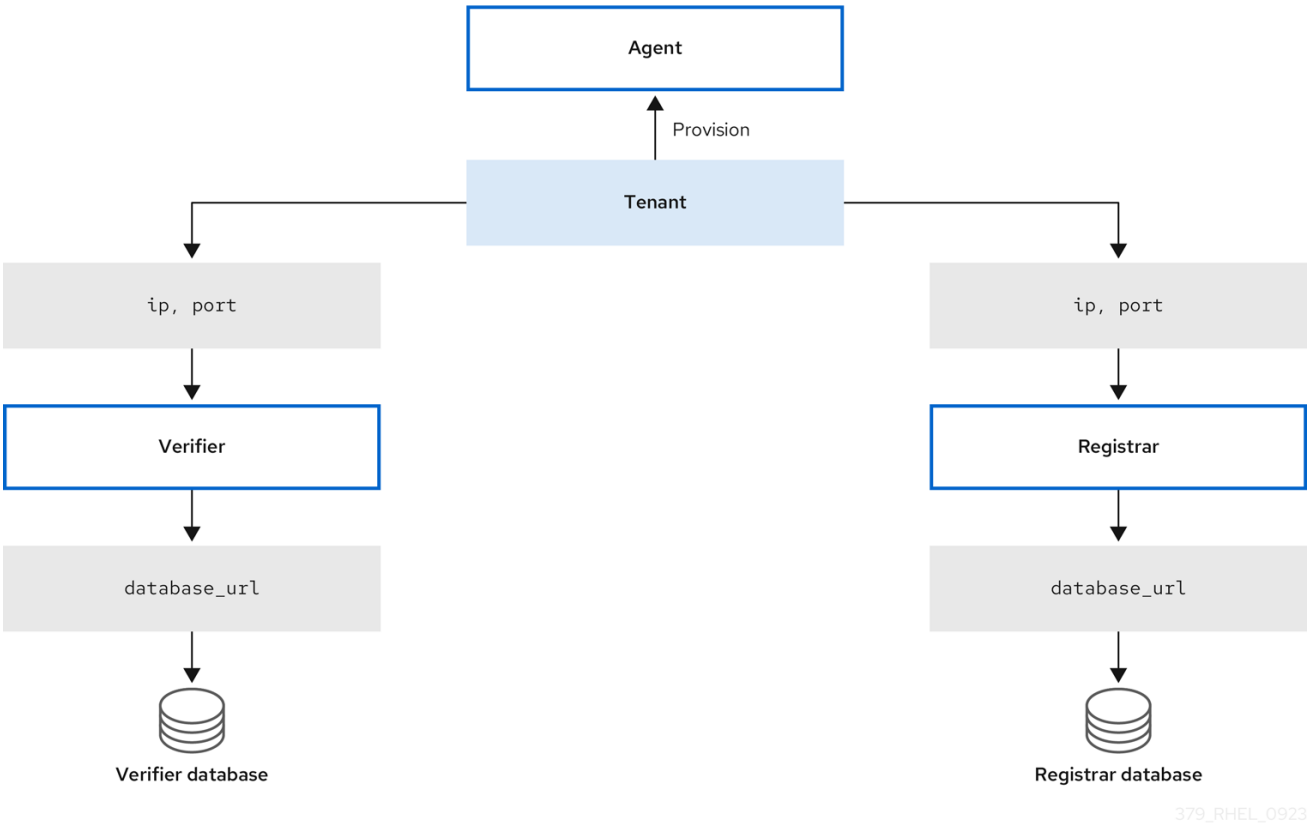
Contains a database of all agents and it hosts the public keys of the TPM vendors.

Agent

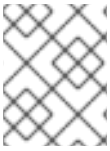
Deployed to remote systems measured by the verifier.

In addition, Keylime uses the **keylime_tenant** utility for many functions, including provisioning the agents on the target systems.

Figure 1.1. Connections between Keylime components through configurations



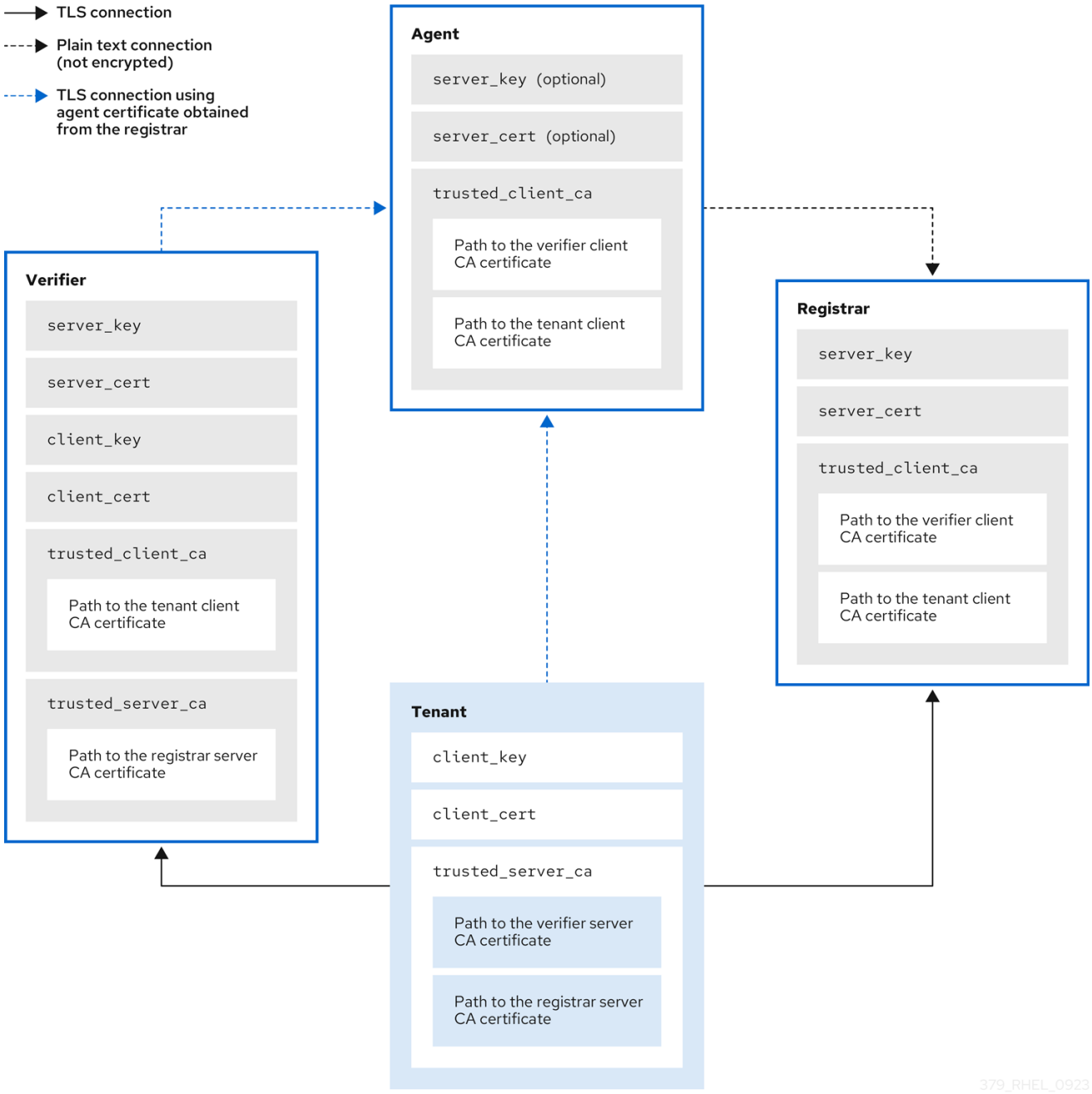
Keylime ensures the integrity of the monitored systems in a chain of trust by using keys and certificates exchanged between the components and the tenant. For a secure foundation of this chain, use a certificate authority (CA) that you can trust.



NOTE

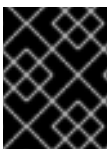
If the agent receives no key and certificate, it generates a key and a self-signed certificate with no involvement from the CA.

Figure 1.2. Connections between Keylime components certificates and keys



1.2. CONFIGURING KEYLIME VERIFIER

The verifier is the most important component in Keylime. It performs initial and periodic checks of system integrity and supports bootstrapping a cryptographic key securely with the agent. The verifier uses mutual TLS encryption for its control interface.



IMPORTANT

To maintain the chain of trust, keep the system that runs the verifier secure and under your control.

You can install the verifier on a separate system or on the same system as the Keylime registrar, depending on your requirements. Running the verifier and registrar on separate systems provides better performance.

NOTE

To keep the configuration files organized within the drop-in directories, use file names with a two-digit number prefix, for example **/etc/keylime/verifier.conf.d/00-verifier-ip.conf**. The configuration processing reads the files inside the drop-in directory in lexicographic order and sets each option to the last value it reads.

Prerequisites

- You have **root** permissions and network connection to the system or systems on which you want to install Keylime components.
- You have valid keys and certificates from your certificate authority.
- Optional: You have access to the databases where Keylime saves data from the verifier. You can use any of the following database management systems:
  - SQLite (default)
  - PostgreSQL
  - MySQL
  - MariaDB

Procedure

1. Install the Keylime verifier:

```
# dnf install keylime-verifier
```

2. Define the IP address and port of verifier by creating a new **.conf** file in the **/etc/keylime/verifier.conf.d/** directory, for example, **/etc/keylime/verifier.conf.d/00-verifier-ip.conf**, with the following content:

```
[verifier]
ip = <verifier_IP_address>
```

   - Replace ***<verifier_IP_address>*** with the verifier's IP address. Alternatively, use **ip = *** or **ip = 0.0.0.0** to bind the verifier to all available IP addresses.
   - Optionally, you can also change the verifier's port from the default value **8881** by using the **port** option.

3. Optional: Configure the verifier's database for the list of agents. The default configuration uses an SQLite database in the verifier's **/var/lib/keylime/cv_data.sqlite/** directory. You can define a different database by creating a new **.conf** file in the **/etc/keylime/verifier.conf.d/** directory, for example, **/etc/keylime/verifier.conf.d/00-db-url.conf**, with the following content:

```
[verifier]
database_url = <protocol>:///<name>:<password>@<ip_address_or_hostname>/<properties>
```

   Replace ***<protocol>:///<name>:<password>@<ip_address_or_hostname>/<properties>*** with the URL of the database, for example, **postgresql:///verifier:UQ?nRNY9g7GZzN7@198.51.100.1/verifierdb**.

Ensure that the credentials you use provide the permissions for Keylime to create the database structure.

4. Add certificates and keys to the verifier. You can either let Keylime generate them, or use existing keys and certificates:

   - With the default **tls_dir = generate** option, Keylime generates new certificates for the verifier, registrar, and tenant in the **/var/lib/keylime/cv_ca/** directory.
   - To load existing keys and certificates in the configuration, define their location in the verifier configuration.

   NOTE

   Certificates must be accessible by the **keylime** user, under which the Keylime services are running.

   Create a new **.conf** file in the **/etc/keylime/verifier.conf.d/** directory, for example, **/etc/keylime/verifier.conf.d/00-keys-and-certs.conf**, with the following content:

   ```
   [verifier]
   tls_dir = /var/lib/keylime/cv_ca
   server_key = </path/to/server_key>
   server_key_password = <passphrase1>
   server_cert = </path/to/server_cert>
   trusted_client_ca = ['</path/to/ca/cert1>', '</path/to/ca/cert2>']
   client_key = </path/to/client_key>
   client_key_password = <passphrase2>
   client_cert = </path/to/client_cert>
   trusted_server_ca = ['</path/to/ca/cert3>', '</path/to/ca/cert4>']
   ```

   NOTE

   Use absolute paths to define key and certificate locations. Alternatively, relative paths are resolved from the directory defined in the **tls_dir** option.

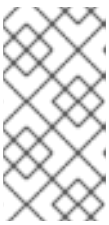
5. Open the port in firewall:

   ```
   # firewall-cmd --add-port 8881/tcp
   # firewall-cmd --runtime-to-permanent
   ```

   If you use a different port, replace **8881** with the port number defined in the **.conf** file.

6. Start the verifier service:

   ```
   # systemctl enable --now keylime_verifier
   ```

   NOTE

   In the default configuration, start the **keylime_verifier** before starting the **keylime_registrar** service because the verifier creates the CA and certificates for the other Keylime components. This order is not necessary when you use custom certificates.

Verification

- Check that the **keylime_verifier** service is active and running:

```
# systemctl status keylime_verifier
● keylime_verifier.service - The Keylime verifier
     Loaded: loaded (/usr/lib/systemd/system/keylime_verifier.service; disabled; vendor preset:
disabled)
     Active: active (running) since Wed 2022-11-09 10:10:08 EST; 1min 45s ago
```

Next steps

- Section 1.4, “Configuring Keylime registrar”.

1.3. CONFIGURING KEYLIME VERIFIER AS A CONTAINER

The Keylime verifier performs initial and periodic checks of system integrity and supports bootstrapping a cryptographic key securely with the agent. You can configure the Keylime verifier as a container instead of the RPM method, without any binaries or packages on the host. The container deployment provides better isolation, modularity, and reproducibility of Keylime components.

After you start the container, the Keylime verifier is deployed with default configuration files. You can customize the configuration by using one or more of following methods:

- Mounting the host’s directories that contain the configuration files to the container. This is available in all versions of RHEL 9.
- Modifying the environment variables directly on the container. This is available in RHEL 9.4 and later versions. Modifying the environment variables overrides the values from the configuration files.

Prerequisites

- The **podman** package and its dependencies are installed on the system.
- Optional: You have access to a database where Keylime saves data from the verifier. You can use any of the following database management systems:
  - SQLite (default)
  - PostgreSQL
  - MySQL
  - MariaDB
- You have valid keys and certificates from your certificate authority.

Procedure

1. Optional: Install the **keylime-verifier** package to access the configuration files. You can configure the container without this package, but it might be easier to modify the configuration files provided with the package.

```
# dnf install keylime-verifier
```

2. Bind the verifier to all available IP addresses by creating a new **.conf** file in the **/etc/keylime/verifier.conf.d/** directory, for example, **/etc/keylime/verifier.conf.d/00-verifier-ip.conf**, with the following content:

```
[verifier]
ip = *
```

   - Optionally, you can also change the verifier's port from the default value **8881** by using the **port** option.

3. Optional: Configure the verifier's database for the list of agents. The default configuration uses an SQLite database in the verifier's **/var/lib/keylime/cv_data.sqlite/** directory. You can define a different database by creating a new **.conf** file in the **/etc/keylime/verifier.conf.d/** directory, for example, **/etc/keylime/verifier.conf.d/00-db-url.conf**, with the following content:

```
[verifier]
database_url = <protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>
```

   Replace ***<protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>*** with the URL of the database, for example, **postgresql://verifier:UQ?nRNY9g7GZzN7@198.51.100.1/verifierdb**.

   Ensure that the credentials you use have the permissions for Keylime to create the database structure.

4. Add certificates and keys to the verifier. You can either let Keylime generate them, or use existing keys and certificates:

   - With the default **tls_dir = generate** option, Keylime generates new certificates for the verifier, registrar, and tenant in the **/var/lib/keylime/cv_ca/** directory.

   - To load existing keys and certificates in the configuration, define their location in the verifier configuration.

NOTE

Certificates must be accessible by the **keylime** user, under which the Keylime processes are running.

Create a new **.conf** file in the **/etc/keylime/verifier.conf.d/** directory, for example, **/etc/keylime/verifier.conf.d/00-keys-and-certs.conf**, with the following content:

```
[verifier]
tls_dir = /var/lib/keylime/cv_ca
server_key = </path/to/server_key>
server_cert = </path/to/server_cert>
trusted_client_ca = ['</path/to/ca/cert1>', '</path/to/ca/cert2>']
client_key = </path/to/client_key>
client_cert = </path/to/client_cert>
trusted_server_ca = ['</path/to/ca/cert3>', '</path/to/ca/cert4>']
```

NOTE

Use absolute paths to define key and certificate locations. Alternatively, relative paths are resolved from the directory defined in the **tls_dir** option.

5. Open the port in firewall:

```
# firewall-cmd --add-port 8881/tcp
# firewall-cmd --runtime-to-permanent
```

If you use a different port, replace **8881** with the port number defined in the **.conf** file.

6. Run the container:

```
$ podman run --name keylime-verifier \
  -p 8881:8881 \
  -v /etc/keylime/verifier.conf.d:/etc/keylime/verifier.conf.d:Z \
  -v /var/lib/keylime/cv_ca:/var/lib/keylime/cv_ca:Z \
  -d \
  -e KEYLIME_VERIFIER_SERVER_KEY_PASSWORD=<passphrase1> \
  -e KEYLIME_VERIFIER_CLIENT_KEY_PASSWORD=<passphrase2> \
  registry.access.redhat.com/rhel9/keylime-verifier
```

- The **-p** option opens the default port **8881** on the host and on the container.
- The **-v** option creates a bind mount for the directory to the container.
  - With the **Z** option, Podman marks the content with a private unshared label. This means only the current container can use the private volume.
- The **-d** option runs the container detached and in the background.
- The option **-e KEYLIME_VERIFIER_SERVER_KEY_PASSWORD=*<passphrase1>*** defines the server key passphrase.
- The option **-e KEYLIME_VERIFIER_CLIENT_KEY_PASSWORD=*<passphrase2>*** defines the client key passphrase.
- You can override configuration options with environment variables by using the option **-e KEYLIME_VERIFIER_*<ENVIRONMENT_VARIABLE>*=*<value>***. To modify additional options, insert the **-e** option separately for each environment variable. For a complete list of environment variables and their default values, see Section 1.12, “Keylime environment variables”.

Verification

- Check that the container is running:

```
$ podman ps -a
CONTAINER ID  IMAGE                                           COMMAND          CREATED
STATUS         PORTS                   NAMES
80b6b9dbf57c  registry.access.redhat.com/rhel9/keylime-verifier:latest  keylime_verifier  14
seconds ago  Up 14 seconds  0.0.0.0:8881->8881/tcp  keylime-verifier
```

Next steps

- Section 1.5, “Configuring Keylime registrar as a container” .

Additional resources

- For more information about Keylime components, see Section 1.1, “How Keylime works”.
- For more information about configuring the Keylime verifier, see Section 1.2, “Configuring Keylime verifier”.
- For more information about the **podman run** command, see the **podman-run(1)** man page.

1.4. CONFIGURING KEYLIME REGISTRAR

The registrar is the Keylime component that contains a database of all agents, and it hosts the public keys of the TPM vendors. After the registrar's HTTPS service accepts trusted platform module (TPM) public keys, it presents an interface to obtain these public keys for checking quotes.

IMPORTANT

To maintain the chain of trust, keep the system that runs the registrar secure and under your control.

You can install the registrar on a separate system or on the same system as the Keylime verifier, depending on your requirements. Running the verifier and registrar on separate systems provides better performance.

NOTE

To keep the configuration files organized within the drop-in directories, use file names with a two-digit number prefix, for example **/etc/keylime/registrar.conf.d/00-registrar-ip.conf**. The configuration processing reads the files inside the drop-in directory in lexicographic order and sets each option to the last value it reads.

Prerequisites

- You have network access to the systems where the Keylime verifier is installed and running. For more information, see Section 1.2, “Configuring Keylime verifier” .
- You have **root** permissions and network connection to the system or systems on which you want to install Keylime components.
- You have access to the database where Keylime saves data from the registrar. You can use any of the following database management systems:
  - SQLite (default)
  - PostgreSQL
  - MySQL
  - MariaDB
- You have valid keys and certificates from your certificate authority.

Procedure

1. Install the Keylime registrar:

```
# dnf install keylime-registrar
```

2. Define the IP address and port of the registrar by creating a new **.conf** file in the **/etc/keylime/registrar.conf.d/** directory, for example, **/etc/keylime/registrar.conf.d/00-registrar-ip.conf**, with the following content:

```
[registrar]
ip = <registrar_IP_address>
```

   - Replace ***<registrar_IP_address>*** with the registrar's IP address. Alternatively, use **ip = *** or **ip = 0.0.0.0** to bind the registrar to all available IP addresses.
   - Optionally, change the port to which the Keylime agents connect by using the **port** option. The default value is **8890**.
   - Optionally, change the TLS port to which the Keylime verifier and tenant connect by using the **tls_port** option. The default value is **8891**.

3. Optional: Configure the registrar's database for the list of agents. The default configuration uses an SQLite database in the registrar's **/var/lib/keylime/reg_data.sqlite** directory. You can create a new **.conf** file in the **/etc/keylime/registrar.conf.d/** directory, for example, **/etc/keylime/registrar.conf.d/00-db-url.conf**, with the following content:

```
[registrar]
database_url = <protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>
```

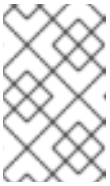
   Replace ***<protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>*** with the URL of the database, for example, **postgresql://registrar:EKYYX-bqY2?#raXm@198.51.100.1/registrardb**.

   Ensure that the credentials you use have the permissions for Keylime to create the database structure.

4. Add certificates and keys to the registrar:

   - You can use the default configuration and load the keys and certificates to the **/var/lib/keylime/reg_ca/** directory.
   - Alternatively, you can define the location of the keys and certificates in the configuration. Create a new **.conf** file in the **/etc/keylime/registrar.conf.d/** directory, for example, **/etc/keylime/registrar.conf.d/00-keys-and-certs.conf**, with the following content:

```
[registrar]
tls_dir = /var/lib/keylime/reg_ca
server_key = </path/to/server_key>
server_key_password = <passphrase1>
server_cert = </path/to/server_cert>
trusted_client_ca = ['</path/to/ca/cert1>', '</path/to/ca/cert2>']
```



NOTE

Use absolute paths to define key and certificate locations. Alternatively, you can define a directory in the **tls_dir** option and use paths relative to that directory.

5. Open the ports in firewall:

```
# firewall-cmd --add-port 8890/tcp --add-port 8891/tcp
# firewall-cmd --runtime-to-permanent
```

If you use a different port, replace **8890** or **8891** with the port number defined in the **.conf** file.

6. Start the **keylime_registrar** service:

```
# systemctl enable --now keylime_registrar
```

NOTE

In the default configuration, start the **keylime_verifier** before starting the **keylime_registrar** service because the verifier creates the CA and certificates for the other Keylime components. This order is not necessary when you use custom certificates.

Verification

- Check that the **keylime_registrar** service is active and running:

```
# systemctl status keylime_registrar
● keylime_registrar.service - The Keylime registrar service
    Loaded: loaded (/usr/lib/systemd/system/keylime_registrar.service; disabled; vendor
preset: disabled)
    Active: active (running) since Wed 2022-11-09 10:10:17 EST; 1min 42s ago
...
```

Next steps

- Section 1.8, “Configuring Keylime tenant”

1.5. CONFIGURING KEYLIME REGISTRAR AS A CONTAINER

The registrar is the Keylime component that contains a database of all agents, and it hosts the public keys of the trusted platform module (TPM) vendors. After the registrar’s HTTPS service accepts TPM public keys, it presents an interface to obtain these public keys for checking quotes. You can configure the Keylime registrar as a container instead of the RPM method, without any binaries or packages on the host. The container deployment provides better isolation, modularity, and reproducibility of Keylime components.

After you start the container, the Keylime registrar is deployed with default configuration files. You can customize the configuration by using one or more of following methods:

- Mounting the host’s directories that contain the configuration files to the container. This is available in all versions of RHEL 9.

- Modifying the environment variables directly on the container. This is available in RHEL 9.4 and later versions. Modifying the environment variables overrides the values from the configuration files.

Prerequisites

- The **podman** package and its dependencies are installed on the system.
- Optional: You have access to a database where Keylime saves data from the registrar. You can use any of the following database management systems:
  - SQLite (default)
  - PostgreSQL
  - MySQL
  - MariaDB
- You have valid keys and certificates from your certificate authority.

Procedure

1. Optional: Install the **keylime-registrar** package to access the configuration files. You can configure the container without this package, but it might be easier to modify the configuration files provided with the package.

```
# dnf install keylime-registrar
```

2. Bind the registrar to all available IP addresses by creating a new **.conf** file in the **/etc/keylime/registrar.conf.d/** directory, for example, **/etc/keylime/registrar.conf.d/00-registrar-ip.conf**, with the following content:

```
[registrar]
ip = *
```

   - Optionally, change the port to which the Keylime agents connect by using the **port** option. The default value is **8890**.
   - Optionally, change the TLS port to which the Keylime tenant connects by using the **tls_port** option. The default value is **8891**.

3. Optional: Configure the registrar's database for the list of agents. The default configuration uses an SQLite database in the registrar's **/var/lib/keylime/reg_data.sqlite** directory. You can create a new **.conf** file in the **/etc/keylime/registrar.conf.d/** directory, for example, **/etc/keylime/registrar.conf.d/00-db-url.conf**, with the following content:

```
[registrar]
database_url = <protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>
```

   Replace ***<protocol>://<name>:<password>@<ip_address_or_hostname>/<properties>*** with the URL of the database, for example, **postgresql://registrar:EKYYX-bqY2?#raXm@198.51.100.1/registrardb**.

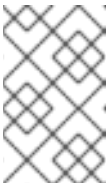
Ensure that the credentials you use have the permissions for Keylime to create the database structure.

4. Add certificates and keys to the registrar:

   - You can use the default configuration and load the keys and certificates to the **/var/lib/keylime/reg_ca/** directory.

   - Alternatively, you can define the location of the keys and certificates in the configuration. Create a new **.conf** file in the **/etc/keylime/registrar.conf.d/** directory, for example, **/etc/keylime/registrar.conf.d/00-keys-and-certs.conf**, with the following content:

```
[registrar]
tls_dir = /var/lib/keylime/reg_ca
server_key = </path/to/server_key>
server_cert = </path/to/server_cert>
trusted_client_ca = ['</path/to/ca/cert1>', '</path/to/ca/cert2>']
```



NOTE

Use absolute paths to define key and certificate locations. Alternatively, you can define a directory in the **tls_dir** option and use paths relative to that directory.

5. Open the ports in firewall:

```
# firewall-cmd --add-port 8890/tcp --add-port 8891/tcp
# firewall-cmd --runtime-to-permanent
```

If you use a different port, replace **8890** or **8891** with the port number defined in the **.conf** file.

6. Run the container:

```
$ podman run --name keylime-registrar \
  -p 8890:8890 \
  -p 8891:8891 \
  -v /etc/keylime/registrar.conf.d:/etc/keylime/registrar.conf.d:Z \
  -v /var/lib/keylime/reg_ca:/var/lib/keylime/reg_ca:Z \
  -d \
  -e KEYLIME_REGISTRAR_SERVER_KEY_PASSWORD=<passphrase1> \
  registry.access.redhat.com/rhel9/keylime-registrar
```

   - The **-p** option opens the default ports **8890** and **8881** on the host and on the container.

   - The **-v** option creates a bind mount for the directory to the container.

     - With the **Z** option, Podman marks the content with a private unshared label. This means only the current container can use the private volume.

   - The **-d** option runs the container detached and in the background.

   - The option **-e KEYLIME_VERIFIER_SERVER_KEY_PASSWORD=*<passphrase1>*** defines the server key passphrase.

   - You can override configuration options with environment variables by using the option **-e**

KEYLIME_REGISTRAR_*<ENVIRONMENT_VARIABLE>=<value>*. To modify additional options, insert the **-e** option separately for each environment variable. For a complete list of environment variables and their default values, see Section 1.12, "Keylime environment variables".

Verification

- Check that the container is running:

```
$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND         CREATED         STATUS
PORTS                                NAMES
07d4b4bff1b6  localhost/keylime-registrar:latest  keylime_registrar  12 seconds ago      Up 12
seconds      0.0.0.0:8881->8881/tcp, 0.0.0.0:8891->8891/tcp  keylime-registrar
```

Next steps

- Section 1.8, "Configuring Keylime tenant".

Additional resources

- For more information about Keylime components, see Section 1.1, "How Keylime works".
- For more information about configuring the Keylime registrar, see Section 1.4, "Configuring Keylime registrar".
- For more information about the **podman run** command, see the **podman-run(1)** man page.

1.6. SETTING UP A KEYLIME SERVER BY USING SYSTEM ROLES

You can set up the verifier and registrar, which are the Keylime server components, by using the **keylime_server** RHEL System Role. The **keylime_server** role installs and configures both the verifier and registrar components together on each node.

Perform this procedure on the Ansible control node.

NOTE

For more information about Keylime, see 8.1. How Keylime works

Prerequisites

- You have prepared the control node and the managed nodes
- You are logged in to the control node as a user who can run playbooks on the managed nodes.
- The account you use to connect to the managed nodes has **sudo** permissions on them.
- The managed nodes or groups of managed nodes on which you want to run this playbook are listed in the Ansible inventory file.

Procedure

1. Create a playbook that defines the required role:

a. Create a new YAML file and open it in a text editor, for example:

```
# vi keylime-playbook.yml
```

b. Insert the following content:

```
---
- name: Manage keylime servers
  hosts: all
  vars:
    keylime_server_verifier_ip: "{{ ansible_host }}"
    keylime_server_registrar_ip: "{{ ansible_host }}"
    keylime_server_verifier_tls_dir: <ver_tls_directory>
    keylime_server_verifier_server_cert: <ver_server_certfile>
    keylime_server_verifier_server_key: <ver_server_key>
    keylime_server_verifier_server_key_passphrase: <ver_server_key_passphrase>
    keylime_server_verifier_trusted_client_ca: <ver_trusted_client_ca_list>
    keylime_server_verifier_client_cert: <ver_client_certfile>
    keylime_server_verifier_client_key: <ver_client_key>
    keylime_server_verifier_client_key_passphrase: <ver_client_key_passphrase>
    keylime_server_verifier_trusted_server_ca: <ver_trusted_server_ca_list>
    keylime_server_registrar_tls_dir: <reg_tls_directory>
    keylime_server_registrar_server_cert: <reg_server_certfile>
    keylime_server_registrar_server_key: <reg_server_key>
    keylime_server_registrar_server_key_passphrase: <reg_server_key_passphrase>
    keylime_server_registrar_trusted_client_ca: <reg_trusted_client_ca_list>
  roles:
    - rhel-system-roles.keylime_server
```

You can find out more about the variables in Variables for the keylime_server RHEL System Role.

2. Run the playbook:

```
$ ansible-playbook <keylime-playbook.yml>
```

Verification

1. Check that the **keylime_verifier** service is active and running on the managed host:

```
# systemctl status keylime_verifier
● keylime_verifier.service - The Keylime verifier
    Loaded: loaded (/usr/lib/systemd/system/keylime_verifier.service; disabled; vendor preset:
disabled)
    Active: active (running) since Wed 2022-11-09 10:10:08 EST; 1min 45s ago
```

2. Check that the **keylime_registrar** service is active and running:

```
# systemctl status keylime_registrar
● keylime_registrar.service - The Keylime registrar service
    Loaded: loaded (/usr/lib/systemd/system/keylime_registrar.service; disabled; vendor
preset: disabled)
    Active: active (running) since Wed 2022-11-09 10:10:17 EST; 1min 42s ago
...
```

Next steps

Section 1.8, “Configuring Keylime tenant”

1.7. VARIABLES FOR THE KEYLIME_SERVER RHEL SYSTEM ROLE

When setting up a Keylime server by using the **keylime_server** RHEL System Role, you can customize the following variables for registrar and verifier.

List of **keylime_server** RHEL System Role variables for configuring the Keylime verifier

keylime_server_verifier_ip

Defines the IP address of the verifier.

keylime_server_verifier_tls_dir

Specifies the directory where the keys and certificates are stored. If set to default, the verifier uses the **/var/lib/keylime/cv_ca** directory.

keylime_server_verifier_server_key_passphrase

Specifies a passphrase to decrypt the server private key. If the value is empty, the private key is not encrypted.

keylime_server_verifier_server_cert: Specifies the Keylime verifier server certificate file.

keylime_server_verifier_trusted_client_ca

Defines the list of trusted client CA certificates. You must store the files in the directory set in the **keylime_server_verifier_tls_dir** option.

keylime_server_verifier_client_key

Defines the file containing the Keylime verifier private client key.

keylime_server_verifier_client_key_passphrase

Defines the passphrase to decrypt the client private key file. If the value is empty, the private key is not encrypted.

keylime_server_verifier_client_cert

Defines the Keylime verifier client certificate file.

keylime_server_verifier_trusted_server_ca

Defines the list of trusted server CA certificates. You must store the files in the directory set in the **keylime_server_verifier_tls_dir** option.

List of registrar variables for setting up keylime_server RHEL System Role

keylime_server_registrar_ip

Defines the IP address of the registrar.

keylime_server_registrar_tls_dir

Specifies the directory where you store the keys and certificates for the registrar. If you set it to default, the registrar uses the **/var/lib/keylime/reg_ca** directory.

keylime_server_registrar_server_key

Defines the Keylime registrar private server key file.

keylime_server_registrar_server_key_passphrase

Specifies the passphrase to decrypt the server private key of the registrar. If the value is empty, the private key is not encrypted.

keylime_server_registrar_server_cert

Specifies the Keylime registrar server certificate file.

keylime_server_registrar_trusted_client_ca

Defines the list of trusted client CA certificates. You must store the files in the directory set in the **keylime_server_registrar_tls_dir** option.

1.8. CONFIGURING KEYLIME TENANT

Keylime uses the **keylime_tenant** utility for many functions, including provisioning the agents on the target systems. You can install **keylime_tenant** on any system, including the systems that run other Keylime components, or on a separate system, depending on your requirements.

Prerequisites

- You have **root** permissions and network connection to the system or systems on which you want to install Keylime components.
- You have network access to the systems where the other Keylime components are configured:

  Verifier

  For more information, see Section 1.2, "Configuring Keylime verifier".

  Registrar

  For more information, see Section 1.4, "Configuring Keylime registrar".

Procedure

1. Install the Keylime tenant:

```
# dnf install keylime-tenant
```

2. Define the tenant's connection to the Keylime verifier by editing the **/etc/keylime/tenant.conf.d/00-verifier-ip.conf** file:

```
[tenant]
verifier_ip = <verifier_ip>
```

   - Replace ***<verifier_ip>*** with the IP address to the verifier's system.
   - If the verifier uses a different port than the default value **8881**, add the **verifier_port = *<verifier_port>*** setting.

3. Define the tenant's connection to the Keylime registrar by editing the **/etc/keylime/tenant.conf.d/00-registrar-ip.conf** file:

```
[tenant]
registrar_ip = <registrar_ip>
registrar_port = <registrar_port>
```

   - Replace ***<registrar_ip>*** with the IP address to the registrar's system.
   - If the registrar uses a different port than the default value **8891**, add the **registrar_port = *<registrar_port>*** setting.

4. Add certificates and keys to the tenant:

   a. You can use the default configuration and load the keys and certificates to the **/var/lib/keylime/cv_ca** directory.

   b. Alternatively, you can define the location of the keys and certificates in the configuration. Create a new **.conf** file in the **/etc/keylime/tenant.conf.d/** directory, for example, **/etc/keylime/tenant.conf.d/00-keys-and-certs.conf**, with the following content:

```
[tenant]
tls_dir = /var/lib/keylime/cv_ca
client_key = tenant-key.pem
client_key_password = <passphrase1>
client_cert = tenant-cert.pem
trusted_server_ca = ['</path/to/ca/cert>']
```

   The **trusted_server_ca** parameter accepts paths to the verifier and registrar server CA certificate. You can provide multiple comma-separated paths, for example if the verifier and registrar use different CAs.

NOTE

Use absolute paths to define key and certificate locations. Alternatively, you can define a directory in the **tls_dir** option and use paths relative to that directory.

5. Optional: If the trusted platform module (TPM) endorsement key (EK) cannot be verified by using certificates in the **/var/lib/keylime/tpm_cert_store** directory, add the certificate to that directory. This can occur particularly when using virtual machines with emulated TPMs.

Verification

1. Check the status of the verifier:

```
# keylime_tenant -c cvstatus
Reading configuration from ['/etc/keylime/logging.conf']
2022-10-14 12:56:08.155 - keylime.tpm - INFO - TPM2-TOOLS Version: 5.2
Reading configuration from ['/etc/keylime/tenant.conf']
2022-10-14 12:56:08.157 - keylime.tenant - INFO - Setting up client TLS...
2022-10-14 12:56:08.158 - keylime.tenant - INFO - Using default client_cert option for tenant
2022-10-14 12:56:08.158 - keylime.tenant - INFO - Using default client_key option for tenant
2022-10-14 12:56:08.178 - keylime.tenant - INFO - TLS is enabled.
2022-10-14 12:56:08.178 - keylime.tenant - WARNING - Using default UUID d432fbb3-d2f1-4a97-9ef7-75bd81c00000
2022-10-14 12:56:08.221 - keylime.tenant - INFO - Verifier at 127.0.0.1 with Port 8881 does not have agent d432fbb3-d2f1-4a97-9ef7-75bd81c00000.
```

   If correctly set up, and if no agent is configured, the verifier responds that it does not recognize the default agent UUID.

2. Check the status of the registrar:

```
# keylime_tenant -c regstatus
Reading configuration from ['/etc/keylime/logging.conf']
2022-10-14 12:56:02.114 - keylime.tpm - INFO - TPM2-TOOLS Version: 5.2
```

```
Reading configuration from ['/etc/keylime/tenant.conf']
2022-10-14 12:56:02.116 - keylime.tenant - INFO - Setting up client TLS...
2022-10-14 12:56:02.116 - keylime.tenant - INFO - Using default client_cert option for tenant
2022-10-14 12:56:02.116 - keylime.tenant - INFO - Using default client_key option for tenant
2022-10-14 12:56:02.137 - keylime.tenant - INFO - TLS is enabled.
2022-10-14 12:56:02.137 - keylime.tenant - WARNING - Using default UUID d432fbb3-d2f1-
4a97-9ef7-75bd81c00000
2022-10-14 12:56:02.171 - keylime.registrar_client - CRITICAL - Error: could not get agent
d432fbb3-d2f1-4a97-9ef7-75bd81c00000 data from Registrar Server: 404
2022-10-14 12:56:02.172 - keylime.registrar_client - CRITICAL - Response code 404: agent
d432fbb3-d2f1-4a97-9ef7-75bd81c00000 not found
2022-10-14 12:56:02.172 - keylime.tenant - INFO - Agent d432fbb3-d2f1-4a97-9ef7-
75bd81c00000 does not exist on the registrar. Please register the agent with the registrar.
2022-10-14 12:56:02.172 - keylime.tenant - INFO - {"code": 404, "status": "Agent d432fbb3-
d2f1-4a97-9ef7-75bd81c00000 does not exist on registrar 127.0.0.1 port 8891.", "results": {}}
```

If correctly set up, and if no agent is configured, the registrar responds that it does not recognize the default agent UUID.

Additional resources

- For additional advanced options for the **keylime_tenant** utility, enter the **keylime_tenant -h** command.

1.9. CONFIGURING KEYLIME AGENT

The Keylime agent is the component deployed to all systems to be monitored by Keylime.

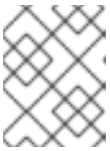
By default, the Keylime agent stores all its data in the **/var/lib/keylime/** directory of the monitored system.

NOTE

To keep the configuration files organized within the drop-in directories, use file names with a two-digit number prefix, for example **/etc/keylime/agent.conf.d/00-registrar-ip.conf**. The configuration processing reads the files inside the drop-in directory in lexicographic order and sets each option to the last value it reads.

Prerequisites

- You have **root** permissions to the monitored system.
- The monitored system has a Trusted Platform Module (TPM).



NOTE

To verify, enter the **tpm2_pcrread** command. If the output returns several hashes, a TPM is available.

- You have network access to the systems where the other Keylime components are configured:

  Verifier

  For more information, see Section 1.2, “Configuring Keylime verifier”.

  Registrar

For more information, see Section 1.4, “Configuring Keylime registrar” .

Tenant

For more information, see Section 1.8, “Configuring Keylime tenant” .

- Integrity measurement architecture (IMA) is enabled on the monitored system. For more information, see Enabling integrity measurement architecture and extended verification module .

Procedure

1. Install the Keylime agent:

```
# dnf install keylime-agent
```

This command installs the **keylime-agent-rust** package.

2. Define the agent's IP address and port in the configuration files. Create a new **.conf** file in the **/etc/keylime/agent.conf.d/** directory, for example, **/etc/keylime/agent.conf.d/00-agent-ip.conf**, with the following content:

```
[agent]
ip = '<agent_ip>'
```

NOTE

The Keylime agent configuration uses the TOML format, which is different from the INI format used for configuration of the other components. Therefore, enter values in valid TOML syntax, for example, paths in single quotation marks and arrays of multiple paths in square brackets.

- Replace ***<agent_IP_address>*** with the agent's IP address. Alternatively, use **ip = '*'** or **ip = '0.0.0.0'** to bind the agent to all available IP addresses.
- Optionally, you can also change the agent's port from the default value **9002** by using the **port = '*<agent_port>*'** option.

3. Define the registrar's IP address and port in the configuration files. Create a new **.conf** file in the **/etc/keylime/agent.conf.d/** directory, for example, **/etc/keylime/agent.conf.d/00-registrar-ip.conf**, with the following content:

```
[agent]
registrar_ip = '<registrar_IP_address>'
```

- Replace ***<registrar_IP_address>*** with the registrar's IP address.
- Optionally, you can also change the registrar's port from the default value **8890** by using the **registrar_port = '*<registrar_port>*'** option.

4. Optional: Define the agent's universally unique identifier (UUID). If it is not defined, the default UUID is used. Create a new **.conf** file in the **/etc/keylime/agent.conf.d/** directory, for example, **/etc/keylime/agent.conf.d/00-agent-uuid.conf**, with the following content:

```
[agent]
uuid = '<agent_UUID>'
```

- Replace ***<agent_UUID>*** with the agent's UUID, for example **d432fbb3-d2f1-4a97-9ef7-abcdef012345**. You can use the **uuidgen** utility to generate a UUID.

5. Optional: Load existing keys and certificates for the agent. If the agent receives no **server_key** and **server_cert**, it generates its own key and a self-signed certificate.
   Define the location of the keys and certificates in the configuration. Create a new **.conf** file in the **/etc/keylime/agent.conf.d/** directory, for example, **/etc/keylime/agent.conf.d/00-keys-and-certs.conf**, with the following content:

```
[agent]
server_key = '</path/to/server_key>'
server_key_password = '<passphrase1>'
server_cert = '</path/to/server_cert>'
trusted_client_ca = '</path/to/ca/cert>'
trusted_client_ca = '[</path/to/ca/cert3>, </path/to/ca/cert4>]'
```

NOTE

Use absolute paths to define key and certificate locations. The Keylime agent does not accept relative paths.

6. Open the port in firewall:

```
# firewall-cmd --add-port 9002/tcp
# firewall-cmd --runtime-to-permanent
```

   If you use a different port, replace **9002** with the port number defined in the **.conf** file.

7. Enable and start the **keylime_agent** service:

```
# systemctl enable --now keylime_agent
```

8. Optional: From the system where the Keylime tenant is configured, verify that the agent is correctly configured and can connect to the registrar.

```
# keylime_tenant -c regstatus --uuid <agent_uuid>
Reading configuration from ['/etc/keylime/logging.conf']
...
==\n-----END CERTIFICATE-----\n", "ip": "127.0.0.1", "port": 9002, "regcount": 1,
"operational_state": "Registered"}}}
```

   - Replace ***<agent_uuid>*** with the agent's UUID.
     If the registrar and agent are correctly configured, the output displays the agent's IP address and port, followed by **"operational_state": "Registered"**.

9. Create a new IMA policy by entering the following content into the **/etc/ima/ima-policy** file:

```
# PROC_SUPER_MAGIC
dont_measure fsmagic=0x9fa0
# SYSFS_MAGIC
dont_measure fsmagic=0x62656572
# DEBUGFS_MAGIC
dont_measure fsmagic=0x64626720
# TMPFS_MAGIC
```

```
dont_measure fsmagic=0x01021994
# RAMFS_MAGIC
dont_measure fsmagic=0x858458f6
# SECURITYFS_MAGIC
dont_measure fsmagic=0x73636673
# SELINUX_MAGIC
dont_measure fsmagic=0xf97cff8c
# CGROUP_SUPER_MAGIC
dont_measure fsmagic=0x27e0eb
# OVERLAYFS_MAGIC
dont_measure fsmagic=0x794c7630
# Do not measure log, audit or tmp files
dont_measure obj_type=var_log_t
dont_measure obj_type=auditd_log_t
dont_measure obj_type=tmp_t
# MEASUREMENTS
measure func=BPRM_CHECK
measure func=FILE_MMAP mask=MAY_EXEC
measure func=MODULE_CHECK uid=0
```

10. Reboot the system to apply the new IMA policy.

Verification

1. Verify that the agent is running:

```
# systemctl status keylime_agent
● keylime_agent.service - The Keylime compute agent
     Loaded: loaded (/usr/lib/systemd/system/keylime_agent.service; enabled; preset:
disabled)
     Active: active (running) since ...
```

Next steps

After the agent is configured on all systems you want to monitor, you can deploy Keylime to perform one or both of the following functions:

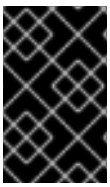
- Section 1.10, “Deploying Keylime for runtime monitoring”
- Section 1.11, “Deploying Keylime for measured boot attestation”

Additional resources

- Integrity Measurement Architecture (IMA) Wiki

1.10. DEPLOYING KEYLIME FOR RUNTIME MONITORING

To verify that the state of monitored systems is correct, the Keylime agent must be running on the monitored systems.



IMPORTANT

Because Keylime runtime monitoring uses integrity measurement architecture (IMA) to measure large numbers of files, it might have a significant impact on the performance of your system.

When provisioning the agent, you can also define a file that Keylime sends to the monitored system. Keylime encrypts the file sent to the agent, and decrypts it only if the agent's system complies with the TPM policy and with the IMA allowlist.

You can make Keylime ignore changes of specific files or within specific directories by configuring a Keylime excludelist.

From Keylime version 7.3.0, provided in RHEL 9.3, the allowlist and excludelist are combined into the Keylime runtime policy.

Prerequisites

- You have network access to the systems where the Keylime components are configured:

  Verifier

  For more information, see Section 1.2, "Configuring Keylime verifier".

  Registrar

  For more information, see Section 1.4, "Configuring Keylime registrar".

  Tenant

  For more information, see Section 1.8, "Configuring Keylime tenant".

  Agent

  For more information, see Section 1.9, "Configuring Keylime agent".

Procedure

1. On the monitored system where the Keylime agent is configured and running, generate an allowlist from the current state of the system:

   ```
   # /usr/share/keylime/scripts/create_allowlist.sh -o <allowlist.txt> -h sha256sum
   ```

   Replace ***<allowlist.txt>*** with the file name of the allowlist.

   IMPORTANT

   Use the SHA-256 hash function. SHA-1 is not secure and has been deprecated in RHEL 9. For additional information, see SHA-1 deprecation in Red Hat Enterprise Linux 9.

2. Copy the generated allowlist to the system where the **keylime_tenant** utility is configured, for example:

   ```
   # scp <allowlist.txt> root@<tenant.ip>:/root/<allowlist.txt>
   ```

3. Optional: You can define a list of files or directories excluded from Keylime measurements by creating a file on the tenant system and entering the files and directories to exclude. The excludelist accepts Python regular expressions with one regular expression per line. See Regular expression operations at docs.python.org for the complete list of special characters. For example, to exclude all files in the **/tmp/** directory and some subdirectories of the **/var/** directory from Keylime measurements, create a **/root/*<excludelist.txt>*** file with the following content:

   ```
   /var/cache/.*
   /var/lock/.*
   ```

```
/var/log/.*
/var/run/.*
/var/spool/.*
/var/tmp/.*
/tmp/.*
```

Save the excludelist on the tenant system.

4. Combine the allowlist and excludelist into the Keylime runtime policy:

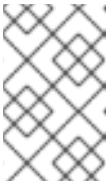
```
# keylime_create_policy -a <allowlist.txt> -e <excludelist.txt> -o <policy.json>
```

5. On the system where the Keylime tenant is configured, provision the agent by using the **keylime_tenant** utility:

```
# keylime_tenant -c add -t <agent_ip> -u <agent_uuid> --runtime-policy <policy.json> --cert
default
```

- Replace ***<agent_ip>*** with the agent's IP address.
- Replace ***<agent_uuid>*** with the agent's UUID.
- Replace ***<policy.json>*** with the path to the Keylime runtime policy file.
- With the **--cert** option, the tenant generates and signs a certificate for the agent by using the CA certificates and keys located in the specified directory, or the default **/var/lib/keylime/ca/** directory. If the directory contains no CA certificates and keys, the tenant will generate them automatically according to the configuration in the **/etc/keylime/ca.conf** file and save them to the specified directory. The tenant then sends these keys and certificates to the agent.
  When generating CA certificates or signing agent certificates, you may be prompted for the password to access the CA private key: **Please enter the password to decrypt your keystore:**.

  If you do not want to use a certificate, use the **-f** option instead for delivering a file to the agent. Provisioning an agent requires sending any file, even an empty file.



NOTE

Keylime encrypts the file sent to the agent, and decrypts it only if the agent's system complies with the TPM policy and the IMA allowlist. By default, Keylime decompresses **.zip** files.

As an example, with the following command, **keylime_tenant** provisions a new Keylime agent at **127.0.0.1** with UUID **d432fbb3-d2f1-4a97-9ef7-75bd81c00000** and loads a runtime policy **policy.json**. It also generates a certificate in the default directory and sends the certificate file to the agent. Keylime decrypts the file only if the TPM policy configured in **/etc/keylime/verifier.conf** is satisfied:

```
# keylime_tenant -c add -t 127.0.0.1 -u d432fbb3-d2f1-4a97-9ef7-75bd81c00000 --cert
default --runtime-policy policy.json
```

NOTE

You can stop Keylime from monitoring a node by using the **# keylime_tenant -c delete -u *<agent_uuid>*** command.

You can modify the configuration of an already registered agent by using the **keylime_tenant -c update** command.

Verification

1. Optional: Reboot the monitored system before verification to verify that the settings are persistent.

2. Verify a successful attestation of the agent:

```
# keylime_tenant -c cvstatus -u <agent.uuid>
...
{"<agent.uuid>": {"operational_state": "Get Quote"..."attestation_count": 5
...
```

Replace ***<agent.uuid>*** with the agent's UUID.

If the value of **operational_state** is **Get Quote** and **attestation_count** is non-zero, the attestation of this agent is successful.

If the value of **operational_state** is **Invalid Quote** or **Failed** attestation fails, the command displays output similar to the following:

```
{"<agent.uuid>": {"operational_state": "Invalid Quote", ... "ima.validation.ima-ng.not_in_allowlist", "attestation_count": 5, "last_received_quote": 1684150329, "last_successful_attestation": 1684150327}}
```

3. If the attestation fails, display more details in the verifier log:

```
# journalctl -u keylime_verifier
keylime.tpm - INFO - Checking IMA measurement list...
keylime.ima - WARNING - File not found in allowlist: /root/bad-script.sh
keylime.ima - ERROR - IMA ERRORS: template-hash 0 fnf 1 hash 0 good 781
keylime.cloudverifier - WARNING - agent D432FBB3-D2F1-4A97-9EF7-75BD81C00000 failed, stopping polling
```

Additional resources

- For more information about IMA, see Enhancing security with the kernel integrity subsystem.

1.11. DEPLOYING KEYLIME FOR MEASURED BOOT ATTESTATION

When you configure Keylime for measured boot attestation, Keylime checks that the boot process on the measured system corresponds to the state you defined.

Prerequisites

- You have network access to the systems where the Keylime components are configured:

Verifier

For more information, see Section 1.2, “Configuring Keylime verifier”.

Registrar

For more information, see Section 1.4, “Configuring Keylime registrar”.

Tenant

For more information, see Section 1.8, “Configuring Keylime tenant”.

Agent

For more information, see Section 1.9, “Configuring Keylime agent”.

- Unified Extensible Firmware Interface (UEFI) is enabled on the agent system.

Procedure

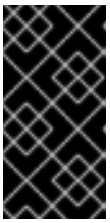
1. On the monitored system where the Keylime agent is configured and running, install the **python3-keylime** package, which contains the **create_mb_refstate** script:

```
# dnf -y install python3-keylime
```

2. On the monitored system, generate a policy from the measured boot log of the current state of the system by using the **create_mb_refstate** script:

```
# /usr/share/keylime/scripts/create_mb_refstate
/sys/kernel/security/tpm0/binary_bios_measurements
<./measured_boot_reference_state.json>
```

Replace ***<./measured_boot_reference_state.json>*** with the path where the script saves the generated policy.



IMPORTANT

The policy generated with the **create_mb_refstate** script is based on the current state of the system and is very strict. Any modifications of the system including kernel updates and system updates will change the boot process and the system will fail the attestation.

3. Copy the generated policy to the system where the **keylime_tenant** utility is configured, for example:

```
# scp root@<agent_ip>:<./measured_boot_reference_state.json>
<./measured_boot_reference_state.json>
```

4. On the system where the Keylime tenant is configured, provision the agent by using the **keylime_tenant** utility:

```
# keylime_tenant -c add -t <agent_ip> -u <agent_uuid> --mb_refstate
<./measured_boot_reference_state.json>
```

- Replace ***<agent_ip>*** with the agent’s IP address.
- Replace ***<agent_uuid>*** with the agent’s UUID.

- Replace ***<./measured_boot_reference_state.json>*** with the path to the measured boot policy.

If you configure measured boot in combination with runtime monitoring, provide all the options from both use cases when entering the **keylime_tenant -c add** command.

NOTE

You can stop Keylime from monitoring a node by using the **# keylime_tenant -c delete -t *<agent_ip>* -u *<agent_uuid>*** command.

You can modify the configuration of an already registered agent by using the **keylime_tenant -c update** command.

Verification

1. Reboot the monitored system and verify a successful attestation of the agent:

```
# keylime_tenant -c cvstatus -u <agent_uuid>
...
{"<agent.uuid>": {"operational_state": "Get Quote"..."attestation_count": 5
...
```

Replace ***<agent_uuid>*** with the agent's UUID.

If the value of **operational_state** is **Get Quote** and **attestation_count** is non-zero, the attestation of this agent is successful.

If the value of **operational_state** is **Invalid Quote** or **Failed** attestation fails, the command displays output similar to the following:

```
{"<agent.uuid>": {"operational_state": "Invalid Quote", ... "ima.validation.ima-ng.not_in_allowlist", "attestation_count": 5, "last_received_quote": 1684150329, "last_successful_attestation": 1684150327}}
```

2. If the attestation fails, display more details in the verifier log:

```
# journalctl -u keylime_verifier
{"d432fbb3-d2f1-4a97-9ef7-75bd81c00000": {"operational_state": "Tenant Quote Failed", ...
"last_event_id": "measured_boot.invalid_pcr_0", "attestation_count": 0,
"last_received_quote": 1684487093, "last_successful_attestation": 0}}
```

1.12. KEYLIME ENVIRONMENT VARIABLES

You can set Keylime environment variables to override the values from the configuration files, for example, when starting a container with the **podman run** command by using the **-e** option.

The environment variables have the following syntax:

```
KEYLIME_<SECTION>_<ENVIRONMENT_VARIABLE>=<value>
```

Where:

- ***<SECTION>*** is the section of the Keylime configuration file.

- ***<ENVIRONMENT_VARIABLE>*** is the environment variable.
- ***<value>*** is the value to which you want to set the environment variable.

For example, **-e KEYLIME_VERIFIER_MAX_RETRIES=6** sets the **max_retries** configuration option in the **[verifier]** section to **6**.

Verifier configuration

Table 1.1. **[verifier]** section

Configuration option	Environment variable	Default value
auto_migrate_db	**KEYLIME_VERIFIER_AUTO_MIGRATE_DB**	**True**
client_cert	**KEYLIME_VERIFIER_CLIENT_CERT**	**default**
client_key_password	**KEYLIME_VERIFIER_CLIENT_KEY_PASSWORD**	
client_key	**KEYLIME_VERIFIER_CLIENT_KEY**	**default**
database_pool_sz_ovfl	**KEYLIME_VERIFIER_DATABASE_POOL_SZ_OVFL**	**5,10**
database_url	**KEYLIME_VERIFIER_DATABASE_URL**	**sqlite**
durable_attestation_import	**KEYLIME_VERIFIER_DURABLE_ATTESTATION_IMPORT**	
enable_agent_mtls	**KEYLIME_VERIFIER_ENABLE_AGENT_MTLS**	**True**
exponential_backoff	**KEYLIME_VERIFIER_EXPONENTIAL_BACKOFF**	**True**
ignore_tomtou_errors	**KEYLIME_VERIFIER_IGNORE_TOMTOU_ERRORS**	**False**
ip	**KEYLIME_VERIFIER_IP**	**127.0.0.1**
max_retries	**KEYLIME_VERIFIER_MAX_RETRIES**	**5**
max_upload_size	**KEYLIME_VERIFIER_MAX_UPLOAD_SIZE**	**104857600**

Configuration option	Environment variable	Default value
measured_boot_evaluate	**KEYLIME_VERIFIER_MEASURED_BOOT_EVALUATE**	**once**
measured_boot_imports	**KEYLIME_VERIFIER_MEASURED_BOOT_IMPORTS**	**[]**
measured_boot_policy_name	**KEYLIME_VERIFIER_MEASURED_BOOT_POLICY_NAME**	**accept-all**
num_workers	**KEYLIME_VERIFIER_NUM_WORKERS**	**0**
persistent_store_encoding	**KEYLIME_VERIFIER_PERSISTENT_STORE_ENCODING**	
persistent_store_format	**KEYLIME_VERIFIER_PERSISTENT_STORE_FORMAT**	**json**
persistent_store_url	**KEYLIME_VERIFIER_PERSISTENT_STORE_URL**	
port	**KEYLIME_VERIFIER_PORT**	**8881**
quote_interval	**KEYLIME_VERIFIER_QUOTE_INTERVAL**	**2**
registrar_ip	**KEYLIME_VERIFIER_REGISTRAR_IP**	**127.0.0.1**
registrar_port	**KEYLIME_VERIFIER_REGISTRAR_PORT**	**8891**
request_timeout	**KEYLIME_VERIFIER_REQUEST_TIMEOUT**	**60.0**
require_allow_list_signatures	**KEYLIME_VERIFIER_REQUIRE_ALLOW_LIST_SIGNATURES**	**True**
retry_interval	**KEYLIME_VERIFIER_RETRY_INTERVAL**	**2**
server_cert	**KEYLIME_VERIFIER_SERVER_CERT**	**default**

Configuration option	Environment variable	Default value
server_key_password	**KEYLIME_VERIFIER_SERVER_KEY_PASSWORD**	**default**
server_key	**KEYLIME_VERIFIER_SERVER_KEY**	**default**
severity_labels	**KEYLIME_VERIFIER_SEVERITY_LABELS**	**["info", "notice", "warning", "error", "critical", "alert", "emergency"]**
severity_policy	**KEYLIME_VERIFIER_SEVERITY_POLICY**	**[{"event_id": ".*", "severity_label" : "emergency"}]**
signed_attributes	**KEYLIME_VERIFIER_SIGNED_ATTRIBUTES**	
time_stamp_authority_certs_path	**KEYLIME_VERIFIER_TIME_STAMP_AUTHORITY_CERTS_PATH**	
time_stamp_authority_url	**KEYLIME_VERIFIER_TIME_STAMP_AUTHORITY_URL**	
tls_dir	**KEYLIME_VERIFIER_TLS_DIR**	**generate**
transparency_log_sign_algo	**KEYLIME_VERIFIER_TRANSPARENCY_LOG_SIGN_ALGO**	**sha256**
transparency_log_url	**KEYLIME_VERIFIER_TRANSPARENCY_LOG_URL**	
trusted_client_ca	**KEYLIME_VERIFIER_TRUSTED_CLIENT_CA**	**default**
trusted_server_ca	**KEYLIME_VERIFIER_TRUSTED_SERVER_CA**	**default**
uuid	**KEYLIME_VERIFIER_UUID**	**default**
version	**KEYLIME_VERIFIER_VERSION**	**2.0**

Table 1.2. **[revocations]** section

Configuration option	Environment variable	Default value
enabled_revocation_notifications	**KEYLIME_VERIFIER_REVOCATIONS_ENABLED_REVOCATION_NOTIFICATIONS**	**[agent]**
webhook_url	**KEYLIME_VERIFIER_REVOCATIONS_WEBHOOK_URL**	

Registrar configuration

Table 1.3. **[registrar]** section

Configuration option	Environment variable	Default value
auto_migrate_db	**KEYLIME_REGISTRAR_AUTO_MIGRATE_DB**	**True**
database_pool_sz_ovfl	**KEYLIME_REGISTRAR_DATABASE_POOL_SZ_OVFL**	**5,10**
database_url	**KEYLIME_REGISTRAR_DATABASE_URL**	**sqlite**
durable_attestation_import	**KEYLIME_REGISTRAR_DURABLE_ATTESTATION_IMPORT**	
ip	**KEYLIME_REGISTRAR_IP**	**127.0.0.1**
persistent_store_encoding	**KEYLIME_REGISTRAR_PERSISTENT_STORE_ENCODING**	
persistent_store_format	**KEYLIME_REGISTRAR_PERSISTENT_STORE_FORMAT**	**json**
persistent_store_url	**KEYLIME_REGISTRAR_PERSISTENT_STORE_URL**	
port	**KEYLIME_REGISTRAR_PORT**	**8890**
prov_db_filename	**KEYLIME_REGISTRAR_PROV_DB_FILENAME**	**provider_reg_data.sqlite**
server_cert	**KEYLIME_REGISTRAR_SERVER_CERT**	**default**

server_key_password	**KEYLIME_REGISTRAR_SERVER_KEY_PASSWORD**	**default**
server_key	**KEYLIME_REGISTRAR_SERVER_KEY**	**default**
signed_attributes	**KEYLIME_REGISTRAR_SIGNED_ATTRIBUTES**	**ek_tpm,aik_tpm,ekcert**
time_stamp_authority_certs_path	**KEYLIME_REGISTRAR_TIME_STAMP_AUTHORITY_CERTS_PATH**	
time_stamp_authority_url	**KEYLIME_REGISTRAR_TIME_STAMP_AUTHORITY_URL**	
tls_dir	**KEYLIME_REGISTRAR_TLS_DIR**	**default**
tls_port	**KEYLIME_REGISTRAR_TLS_PORT**	**8891**
transparency_log_sign_algo	**KEYLIME_REGISTRAR_TRANSPARENCY_LOG_SIGN_ALGO**	**sha256**
transparency_log_url	**KEYLIME_REGISTRAR_TRANSPARENCY_LOG_URL**	
trusted_client_ca	**KEYLIME_REGISTRAR_TRUSTED_CLIENT_CA**	**default**
version	**KEYLIME_REGISTRAR_VERSION**	**2.0**

Tenant configuration

Table 1.4. **[tenant]** section

Configuration option	Environment variable	Default value
accept_tpm_encryption_algs	**KEYLIME_TENANT_ACCEPT_TPM_ENCRYPTION_ALGS**	**ecc, rsa**
accept_tpm_hash_algs	**KEYLIME_TENANT_ACCEPT_TPM_HASH_ALGS**	**sha512, sha384, sha256**

accept_tpm_signing_algs	KEYLIME_TENANT_ACCEPT_TPM_SIGNING_ALGS	ecschnorr, rsassa
client_cert	KEYLIME_TENANT_CLIENT_CERT	default
client_key_password	KEYLIME_TENANT_CLIENT_KEY_PASSWORD	
client_key	KEYLIME_TENANT_CLIENT_KEY	default
ek_check_script	KEYLIME_TENANT_EK_CHECK_SCRIPT	
enable_agent_mtls	KEYLIME_TENANT_ENABLE_AGENT_MTLS	True
exponential_backoff	KEYLIME_TENANT_EXPONENTIAL_BACKOFF	True
max_payload_size	KEYLIME_TENANT_MAX_PAYLOAD_SIZE	1048576
max_retries	KEYLIME_TENANT_MAX_RETRIES	5
mb_refstate	KEYLIME_TENANT_MB_REFSTATE	
registrar_ip	KEYLIME_TENANT_REGISTRAR_IP	127.0.0.1
registrar_port	KEYLIME_TENANT_REGISTRAR_PORT	8891
request_timeout	KEYLIME_TENANT_REQUEST_TIMEOUT	60
require_ek_cert	KEYLIME_TENANT_REQUIRE_EK_CERT	True
retry_interval	KEYLIME_TENANT_RETRY_INTERVAL	2
tls_dir	KEYLIME_TENANT_TLS_DIR	default
tpm_cert_store	KEYLIME_TENANT_TPM_CERT_STORE	/var/lib/keylime/tpm_cert_store

trusted_server_ca	**KEYLIME_TENANT_TRUSTED_SERVER_CA**	**default**
verifier_ip	**KEYLIME_TENANT_VERIFIER_IP**	**127.0.0.1**
verifier_port	**KEYLIME_TENANT_VERIFIER_PORT**	**8881**
version	**KEYLIME_TENANT_VERSION**	**2.0**

CA configuration

Table 1.5. **[ca]** section

Configuration option	Environment variable	Default value
cert_bits	**KEYLIME_CA_CERT_BITS**	**2048**
cert_ca_lifetime	**KEYLIME_CA_CERT_CA_LIFETIME**	**3650**
cert_ca_name	**KEYLIME_CA_CERT_CA_NAME**	**Keylime Certificate Authority**
cert_country	**KEYLIME_CA_CERT_COUNTRY**	**US**
cert_crl_dist	**KEYLIME_CA_CERT_CRL_DIST**	**http://localhost:38080/crl**
cert_lifetime	**KEYLIME_CA_CERT_LIFETIME**	**365**
cert_locality	**KEYLIME_CA_CERT_LOCALITY**	**Lexington**
cert_org_unit	**KEYLIME_CA_CERT_ORG_UNIT**	**53**
cert_organization	**KEYLIME_CA_CERT_ORGANIZATION**	**MITLL**
cert_state	**KEYLIME_CA_CERT_STATE**	**MA**
password	**KEYLIME_CA_PASSWORD**	**default**

version	**KEYLIME_CA_VERSION**	**2.0**

Agent configuration

Table 1.6. **[agent]** section

Configuration option	Environment variable	Default value
contact_ip	**KEYLIME_AGENT_CONTACT_IP**	**127.0.0.1**
contact_port	**KEYLIME_AGENT_CONTACT_PORT**	**9002**
dec_payload_file	**KEYLIME_AGENT_DEC_PAYLOAD_FILE**	**decrypted_payload**
ek_handle	**KEYLIME_AGENT_EK_HANDLE**	**generate**
enable_agent_mtls	**KEYLIME_AGENT_ENABLE_AGENT_MTLS**	**true**
enable_insecure_payload	**KEYLIME_AGENT_ENABLE_INSECURE_PAYLOAD**	**false**
enable_revocation_notifications	**KEYLIME_AGENT_ENABLE_REVOCATION_NOTIFICATIONS**	**true**
enc_keyname	**KEYLIME_AGENT_ENC_KEYNAME**	**derived_tci_key**
exponential_backoff	**KEYLIME_AGENT_EXPONENTIAL_BACKOFF**	**true**
extract_payload_zip	**KEYLIME_AGENT_EXTRACT_PAYLOAD_ZIP**	**true**
ip	**KEYLIME_AGENT_IP**	**127.0.0.1**
max_retries	**KEYLIME_AGENT_MAX_RETRIES**	**4**
measure_payload_pcr	**KEYLIME_AGENT_MEASURE_PAYLOAD_PCR**	**-1**
payload_script	**KEYLIME_AGENT_PAYLOAD_SCRIPT**	**autorun.sh**

Configuration option	Environment variable	Default value
port	KEYLIME_AGENT_PORT	9002
registrar_ip	KEYLIME_AGENT_REGISTRAR_IP	127.0.0.1
registrar_port	KEYLIME_AGENT_REGISTRAR_PORT	8890
retry_interval	KEYLIME_AGENT_RETRY_INTERVAL	2
revocation_actions	KEYLIME_AGENT_REVOCATION_ACTIONS	[]
revocation_cert	KEYLIME_AGENT_REVOCATION_CERT	default
revocation_notification_ip	KEYLIME_AGENT_REVOCATION_NOTIFICATION_IP	127.0.0.1
revocation_notification_port	KEYLIME_AGENT_REVOCATION_NOTIFICATION_PORT	8992
run_as	KEYLIME_AGENT_RUN_AS	keylime:tss
secure_size	KEYLIME_AGENT_SECURE_SIZE	1m
server_cert	KEYLIME_AGENT_SERVER_CERT	default
server_key_password	KEYLIME_AGENT_SERVER_KEY_PASSWORD	
server_key	KEYLIME_AGENT_SERVER_KEY	default
tls_dir	KEYLIME_AGENT_TLS_DIR	default
tpm_encryption_alg	KEYLIME_AGENT_TPM_ENCRYPTION_ALG	rsa
tpm_hash_alg	KEYLIME_AGENT_TPM_HASH_ALG	sha256

Configuration option	Environment variable	Default value
tpm_ownerpassword	**KEYLIME_AGENT_TPM_OWNERPASSWORD**	
tpm_signing_alg	**KEYLIME_AGENT_TPM_SIGNING_ALG**	**rsassa**
trusted_client_ca	**KEYLIME_AGENT_TRUSTED_CLIENT_CA**	**default**
uuid	**KEYLIME_AGENT_UUID**	**d432fbb3-d2f1-4a97-9ef7-75bd81c00000**
version	**KEYLIME_AGENT_VERSION**	**2.0**

Logging configuration

Table 1.7. **[logging]** section

Configuration option	Environment variable	Default value
version	**KEYLIME_LOGGING_VERSION**	**2.0**

Table 1.8. **[loggers]** section

Configuration option	Environment variable	Default value
keys	**KEYLIME_LOGGING_LOGGERS_KEYS**	**root,keylime**

Table 1.9. **[handlers]** section

Configuration option	Environment variable	Default value
keys	**KEYLIME_LOGGING_HANDLERS_KEYS**	**consoleHandler**

Table 1.10. **[formatters]** section

Configuration option	Environment variable	Default value
keys	**KEYLIME_LOGGING_FORMATTERS_KEYS**	**formatter**

Table 1.11. **[formatter_formatter]** section

Configuration option	Environment variable	Default value
datefmt	**KEYLIME_LOGGING_FORMATTER_FORMATTER_DATEFMT**	**%Y-%m-%d %H:%M:%S**
format	**KEYLIME_LOGGING_FORMATTER_FORMATTER_FORMAT**	**%(asctime)s.%(msecs)03d - %(name)s - %(levelname)s - %(message)s**

Table 1.12. **[logger_root]** section

Configuration option	Environment variable	Default value
handlers	**KEYLIME_LOGGING_LOGGER_ROOT_HANDLERS**	**consoleHandler**
level	**KEYLIME_LOGGING_LOGGER_ROOT_LEVEL**	**INFO**

Table 1.13. **[handler_consoleHandler]** section

Configuration option	Environment variable	Default value
args	**KEYLIME_LOGGING_HANDLER_CONSOLEHANDLER_ARGS**	**(sys.stdout,)**
class	**KEYLIME_LOGGING_HANDLER_CONSOLEHANDLER_CLASS**	**StreamHandler**
formatter	**KEYLIME_LOGGING_HANDLER_CONSOLEHANDLER_FORMATTER**	**formatter**
level	**KEYLIME_LOGGING_HANDLER_CONSOLEHANDLER_LEVEL**	**INFO**

Table 1.14. **[logger_keylime]** section

Configuration option	Environment variable	Default value
handlers	**KEYLIME_LOGGING_LOGGER_KEYLIME_HANDLERS**	

level	**KEYLIME_LOGGING_LOGGER_KEYLIME_LEVEL**	**INFO**
qualname	**KEYLIME_LOGGING_LOGGER_KEYLIME_QUALNAME**	**keylime**