Keylime Documentation Documentation

Release 3.0.0

Keylime Developers

Nov 27, 2019

Contents:

Warning: This documentation is still under development and not complete. It will be so until this warning is removed.

Welcome to the Keylime Documentation site!

Keylime is a TPM-based highly scalable remote boot attestation and runtime integrity measurement solution. Keylime enables cloud users to monitor remote nodes using a hardware based cryptographic root of trust.

Keylime was originally born out of the security research team in MIT's Lincoln Laboratory and is now developed and maintained by the Keylime community.

This Documentation site contains guides to install, use and administer keylime as well as guides to enable developers to make contributions to keylime or develop services against Keylime's Rest API(s).

CHAPTER 1

Installation

There are three current methods for installing Keylime, the ansible role, the keylime installer or a manual installation.

1.1 Ansible Keylime Roles

An Ansible role to deploy Keylime , alongside the Keylime rust cloud agent

> **Warning:** Please note that the rust cloud agent is still under early stages of Development. Those wishing to test drive keylimes functionality should use the existing python based cloud agent *keylime_agent* until later notice.

This role deploys keylime for use with a Hardware TPM.

Should you wish to deploy Keylime with a software TPM emulator for development or getting your feet wet, use the Ansible Keylime Soft TPM role instead.

1.1.1 Usage

Download or clone Ansible Keylime from its repository and follow the usage section.

Run the example playbook against your target remote host(s):

```
ansible-playbook -i your_hosts playbook.yml
```

1.1.2 TPM Version Control (Software TPM)

Ansible Keylime Soft TPM provides two role types for both 1.2 and 2.0 TPM versions.

Either TPM version 1.2 or TPM 2.0 support can be configured by simply changing the role in the *playbook.yml* file here

For TPM 2.0 use:

```
- ansible-keylime-tpm20
```

For TPM 1.20 use:

```
- ansible-keylime-tpm12
```

Both roles will deploy the relevant TPM 1.2 Emulator (tpm4720) or 2.0 Emulator (IBM software TPM).

1.1.3 Vagrant

If you prefer, a *Vagrantfile* is available for provisioning.

Clone the repository and then simply run:

```
vagrant up --provider <provider> --provision
```

For example, using libvirt:

```
vagrant up --provider libvirt --provision
```

For example, using VirtualBox:

```
vagrant up --provider virtualbox --provision
```

Once the VM is started, vagrant ssh into the VM and run *sudo su* - to become root.

You can then start the various components using commands:

```
keylime_verifier
keylime_registrar
keylime_agent
```

1.1.4 WebApp

The web application can be started with the command *keylime_webapp*. If using Vagrant, port 443 will be forwarded from the guest to port 8443 on the host.

This will result in the web application being available on url:

https://localhost:8443/webapp/

1.1.5 Rust Cloud agent

To start the rust cloud agent, navigate to it's repository directory and use cargo to run:

```
[root@localhost rust-keylime]# RUST_LOG=keylime_agent=trace cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.28s
    Running `target/debug/keylime_agent`
    INFO  keylime_agent > Starting server...
    INFO  keylime_agent > Listening on http://127.0.0.1:1337
```

1.2 Keylime Bash installer

Keylime requires Python 2.7.10 or newer for proper TLS support.

Installation can be performed via an automated shell script, *installer.sh*. The following command line options are available:

```
Usage: ./installer.sh [option...]
Options:
-k              Download Keylime (stub installer mode)
-o              Use OpenSSL instead of CFSSL
-t              Create tarball with keylime_agent
-m              Use modern TPM 2.0 libraries (vs. TPM 1.2)
-s              Install TPM in socket/simulator mode (vs. chardev)
-p PATH         Use PATH as Keylime path
-h              This help info
```

Note that CFSSL is required if you want to support revocation. As noted above, do not use the TPM emulator option *-s* in production systems.

1.3 Docker (Development Only)

Python keylime and related emulators can also be deployed using Docker. Since this docker configuration currently uses a TPM emulator, it should only be used for development or testing and NOT in production.

Please see either the Dockerfiles here or our local CI script here which will automate the build and pull of keylime on TPM 1.2 or 2.0.

1.4 Manual

Keylime requires Python 2.7.10 or newer for proper TLS support. This is newer than some LTS distributions like Ubuntu 14.04 or CentOS 7. See google for instructions on how to get a newer Python onto those platforms.

1.4.1 Python-based prerequisites

Note: The following outlines installing Keylime under the Python 2 environment, work is underway to port Keylime to Python 3.

The following python packages are required:

- pycryptodomex>=3.4.1
- tornado>=4.3
- m2crypto>=0.21.1
- pyzmq>=14.4
- setuptools>=0.7
- python-dev
- pyyaml

The latter of these are usually available as distro packages. See installer.sh for more information if you want to install them this way. You can also let keylime's *setup.py* install them via PyPI.

1.4.2 TPM 1.2 Support

You also need a patched version of tpm4720 the IBM software TPM emulator and utilities. This is available here Even if you are using keylime with a real TPM, you must install the IBM emulator because keylime uses the command line utilities that come with it. See README.md in that project for detailed instructions on how to build and install it.

The brief synopsis of a quick build/install (after installing dependencies) is:

```
git clone https://github.com/keylime/tpm4720-keylime.git
cd tpm4720-keylime/libtpm
./comp-chardev.sh
sudo make install
```

To build tpm4720 to use the TPM emulator replace *./comp-chardev.sh* with *./comp-sockets.sh*.

To ensure that you have the patched version installed ensure that you have the *encaik* utility in your path.

1.4.3 TPM 2.0 Support

Keylime uses the Intel TPM2 software set to provide TPM 2.0 support. You will need to install the tpm2-tss software stack (available here) as well as a patched version of the tpm2-tools utilities available **'here<https://github.com/keylime/tpm2-tools>'_**. See README.md in these projects for detailed instructions on how to build and install.

The brief synopsis of a quick build/install (after installing dependencies) is:

```
# tpm2-tss
git clone https://github.com/tpm2-software/tpm2-tss.git tpm2-tss
pushd tpm2-tss
./bootstrap
./configure --prefix=/usr
make
sudo make install
popd
# tpm2-tools
git clone https://github.com/keylime/tpm2-tools.git tpm2-tools
pushd tpm2-tools
./bootstrap
./configure --prefix=/usr/local
make
sudo make install
```

To ensure that you have the patched version installed ensure that you have the *tpm2_checkquote* utility in your path.

1.4.4 TPM 2.0 Resource Manager

Note that it is recommended that you use the tpm2-abrmd resource manager (available at https://github.com/tpm2-software/tpm2-abrmd) as well instead of communicating directly with the TPM. See README.md at that project for detailed instructions on how to build and install.

A brief, workable example for Ubuntu 18 LTS systems is:

```
sudo useradd --system --user-group tss
git clone https://github.com/tpm2-software/tpm2-abrmd.git tpm2-abrmd
pushd tpm2-abrmd
./bootstrap
./configure --with-dbuspolicydir=/etc/dbus-1/system.d \
            --with-systemdsystemunitdir=/lib/systemd/system \
            --with-systemdpresetdir=/lib/systemd/system-preset \
            --datarootdir=/usr/share
make
sudo make install
sudo ldconfig
sudo pkill -HUP dbus-daemon
sudo systemctl daemon-reload
sudo service tpm2-abrmd start
export TPM2TOOLS_TCTI="tabrmd:bus_name=com.intel.tss2.Tabrmd"
```

NOTE: if using swtpm2 emulator, you need to run the tpm2-abrmd service as:

```
sudo -u tss /usr/local/sbin/tpm2-abrmd --tcti=mssim &
```

Alternatively, it is also possible, though not recommended, to communicate directly with the TPM (and not use a resource manager). This can be done by setting the environment var *TPM2TOOLS_TCTI* to the appropriate value:

To talk directly to the swtpm2 emulator:

```
export TPM2TOOLS_TCTI="mssim:port=2321"`
```

To talk directly to a real TPM:

```
export TPM2TOOLS_TCTI="device:/dev/tpm0"
```

1.4.5 Install Keylime

You're finally ready to install keylime:

```
sudo python setup.py install
```

1.4.6 To run on OSX 10.11+

You need to build m2crypto from source with:

```
brew install openssl
git clone https://gitlab.com/m2crypto/m2crypto.git
python setup.py build build_ext --openssl=/usr/local/opt/openssl/
sudo -E python setup.py install build_ext --openssl=/usr/local/opt/openssl/
```

1.4.7 Optional Requirements

If you want to support revocation, you also need to have cfssl installed and in your path on the tenant agent. It can be obtained from here. You will also need to set ca_implementation to "cfssl" instead of "openssl" in */etc/keylime.conf*.

CHAPTER 2

User Guide

2.1 User Selected PCR Monitoring

Warning: This page is still under development and not complete. It will be so until this warning is removed.

Using use the *tpm_policy* feature in Keylime, it is possible to mointor a remote machine for any given PCR.

This can be used for Trusted Boot checks for both the *rhboot* shim loader and Trusted Grub 2.

2.1.1 How to use

Select which PCRs you would like Keylime to measure, by using the *tpm2_pcrlist* tool.

Now you can set the PCR values as an array in either the *keylime.conf* file:

```
tpm_policy = {"22":["0000000000000000000000000000000000000001",
↪"0000000000000000000000000000000000000000000000000000000000000001",
↪"0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
↪","ffffffffffffffffffffffffffffffffffffffff",
↪"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
↪"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
↪"],"15":["0000000000000000000000000000000000000000",
↪"0000000000000000000000000000000000000000000000000000000000000000",
↪"0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪"]}
```

Or you can add a node to using *keylime_tenant*:

```
keylime_tenant -v 127.0.0.1 -t 127.0.0.1 -f /root/excludes.txt \
--uuid D432FBB3-D2F1-4A97-9EF7-75BD81C00000 \
--whitelist /root/whitelist.txt \
```

(continues on next page)

(continued from previous page)

```
--exclude /root/exclude.txt \
--tpm_policy  {"22":["0000000000000000000000000000000000000001",
↪"0000000000000000000000000000000000000000000000000000000000000001",
↪"0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
↪","ffffffffffffffffffffffffffffffffffffffff",
↪"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
↪"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
↪"],"15":["0000000000000000000000000000000000000000",
↪"0000000000000000000000000000000000000000000000000000000000000000",
↪"0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪"]} \
-c add
```

2.1.2 rhboot shim-loader

The following is sourced from the *rhboot shim repository <https://github.com/rhboot/shim/blob/master/README.tpm>_* please visit the upstream README to ensure information is still accurate

The following PCRs are extended by shim:

PCR4: - the Authenticode hash of the binary being loaded will be extended into

PCR4 before SB verification.

- the hash of any binary for which Verify is called through the shim_lock protocol

PCR7: - Any certificate in one of our certificate databases that matches a binary

we try to load will be extended into PCR7. That includes: - DBX - the system blacklist, logged as “dbx” - MokListX - the Mok blacklist, logged as “MokListX” - vendor_dbx - shim’s built-in vendor blacklist, logged as “dbx” - DB - the system whitelist, logged as “db” - MokList the Mok whitelist, logged as “MokList” - vendor_cert - shim’s built-in vendor whitelist, logged as “Shim” - shim_cert - shim’s build-time generated whitelist, logged as “Shim”

- MokSBState will be extended into PCR7 if it is set, logged as “MokSBState”.

PCR8: - If you’re using the grub2 TPM patchset we cary in Fedora, the kernel command

line and all grub commands (including all of grub.cfg that gets run) are measured into PCR8.

PCR9: - If you’re using the grub2 TPM patchset we cary in Fedora, the kernel,

initramfs, and any multiboot modules loaded are measured into PCR9.

PCR14: - MokList, MokListX, and MokSBState will be extended into PCR14 if they are

set.

e

2.2 Run-time Integrity Monitoring

Keylimes run-time integrity monitoring requires the set up of Linux IMA.

You should refer to your Linux Distributions documentation to enable IMA, but as a general guide most recent versions already have *CONFIG_IMA* toggled to *Y* as a value during Kernel compile.

It is then just a case of deploying an *ima-policy* file. On a Fedora or Debian system, the file is situated in */etc/ima/ima-policy*.

For configuration of your IMA policy, please refer to the IMA Documentation

Within Keylime we use the following for demonstration:

```
# 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
# MEASUREMENTS
measure func=BPRM_CHECK
measure func=FILE_MMAP mask=MAY_EXEC
measure func=MODULE_CHECK uid=0
```

This default policy measures all executables in *bprm_check*, all files *mmapped* executable in *file_mmap* and module checks.

Once your *ima-policy* is in place, reboot your machine (or even better have it present in your image for first boot).

You can then verify IMA is measuring your system:

```
# head -5 /sys/kernel/security/ima/ascii_runtime_measurements
PCR                                template-hash filedata-hash
↪           filename-hint
10 3c93cea361cd6892bc8b9e3458e22ce60ef2e632 ima-ng␣
↪sha1:ac7dd11bf0e3bec9a7eb2c01e495072962fb9dfa boot_aggregate
10 3d1452eb1fcbe51ad137f3fc21d3cf4a7c2e625b ima-ng␣
↪sha1:a212d835ca43d7deedd4ee806898e77eab53dafa /usr/lib/systemd/systemd
10 e213099a2bf6d88333446c5da617e327696f9eb4 ima-ng␣
↪sha1:6da34b1b7d2ca0d5ca19e68119c262556a15171d /usr/lib64/ld-2.28.so
10 7efd8e2a3da367f2de74b26b84f20b37c692b9f9 ima-ng␣
↪sha1:af78ea0b455f654e9237e2086971f367b6bebc5f /usr/lib/systemd/libsystemd-shared-
↪239.so
10 784fbf69b54c99d4ae82c0be5fca365a8272414e ima-ng␣
↪sha1:b0c601bf82d32ff9afa34bccbb7e8f052c48d64e /etc/ld.so.cache
```

2.2.1 Keylime IMA whitelists

A whitelist is a set of "golden" cryptographic hashes of a files un-tampered state.

The structure of the white list is a hash followed by a full POSIX path to the file:

```
ffe3ad4c395985d143bd0e45a9a1dd09aac21b91 /path/to/file
```

Keylime will load the whitelist into the Keylime Verifier. Keylime will then poll tpm quotes to *PCR 10* on the agents TPM and validate the agents file(s) state against the whitelist. If the object has been tampered with, the hashes will not match and Keylime will place the agent into a failed state. Likewise, if any files invoke the actions stated in *ima-policy* that are not matched in the whitelist, keylime will place the agent into a failed state.

Generate a whitelist

Keylime provides a script to generate whitelists from *initramfs*, but this is only a guide. We encourage developers / users of Keylime to be creative and come up with their own process for securely creating and maintaining a whitelist.

The *create_whitelist.sh* script is available here

Run the script as follows:

```
# create_whitelist.sh  whitelist.txt [hash-algo]
```

With *[hash-algo]* being *sha1sum*, *sha256sum* (note, you need the OpenSSL app installed to have the shasum CLI applications available).

This will then result in *whitelist.txt* being available for Agent provisioning.

Warning: It's best practice to create the whitelist in a secure environment. Ideally, this should be on a fully encrypted, air gapped computer that is permanently isolated from the Internet. Disable all network cards and sign the whitelist hash to ensure no tampering occurs when transferring to other machines.

Alongside building a whitelist from *initramfs*, you could also generate good hashes for your applications files or admin scripts that will run on the remotely attested machine.

Excludes List

An excludes list can be utilised to exclude any file or path. The excludes list supports standard regular expressions, for example the *tmp* directory can be ignored:

```
/tmp/*
```

Remotely Provision Agents

Now that we have our whitelist available, we can send it to the verifier.

Note: If you're using a TPM Emulator (for example with the ansible-keylime-tpm-emulator, you will also need to run the keylime ima emulator. To do this, open a terminal and run *keylime_ima_emulator*

Using the *keylime_tenant* we can send the whitelist and our excludes list as follows:

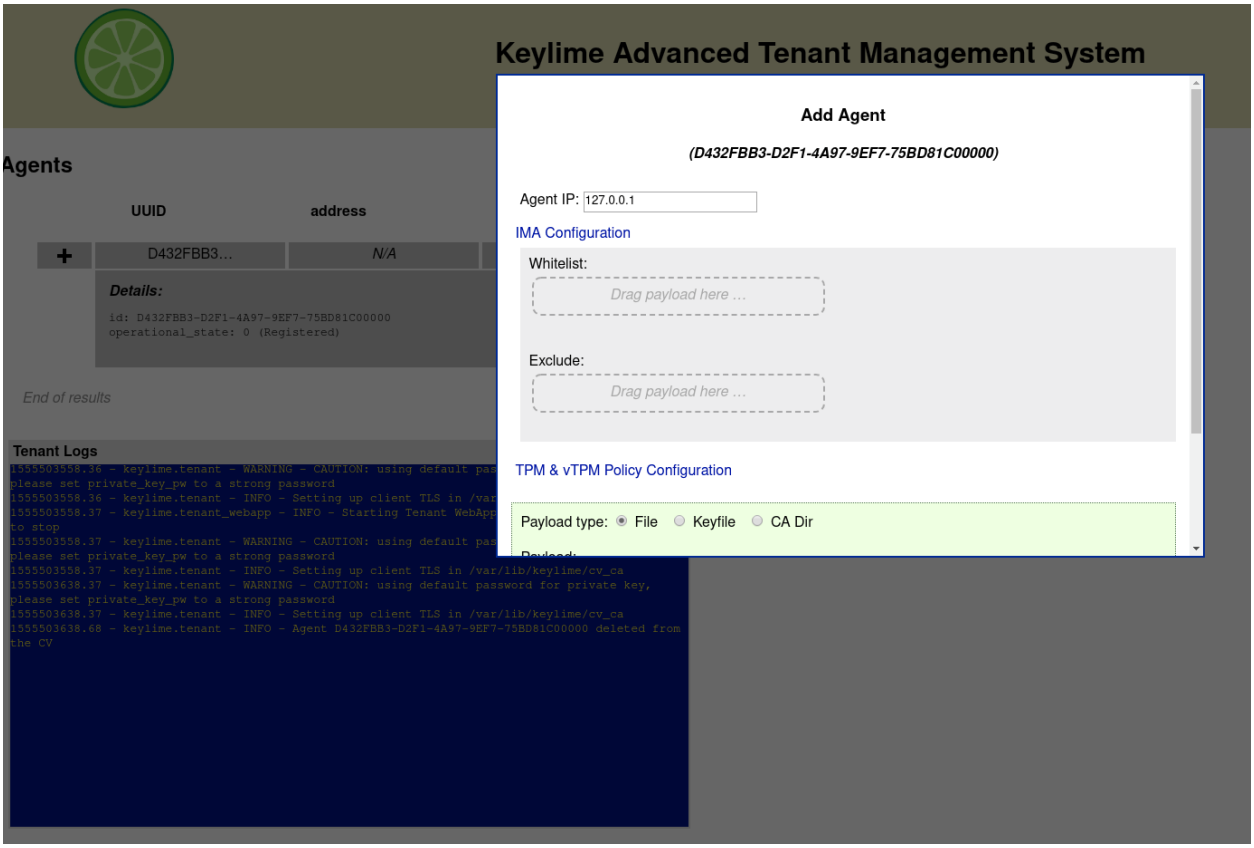
```
keylime_tenant -v <verifier-ip> -t <agent-ip> -f /path/excludes.txt --uuid D432FBB3-
↪D2F1-4A97-9EF7-75BD81C00000 --whitelist /path/whitelist.txt --exclude /path/
↪excludes.txt
```

Note: If your agent is already registered, you can use *-c update*

Should you prefer, you can set the values *ima_whitelist* & *ima_excludelist* within */etc/keylime.conf*, you can then use *default* as follows:

```
`keylime_tenant -v 127.0.0.1 -t neptune -f /root/excludes.txt --uuid D432FBB3-D2F1-
↪4A97-9EF7-75BD81C00000 --whitelist default --exclude default`
```

The whitelist can also be uploaded using the WebApp:



2.2.2 How can I test this?

Create a script that does anything (for example *echo "hello world"*) that is not present in your whitelist or the excludes list. Run the script as root on the agent machine. You will then see the following output on the verifier showing the agent status change to failed:

```
keylime.tpm - INFO - Checking IMA measurement list...
keylime.ima - WARNING - File not found in whitelist: /root/evil_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
```

2.3 Encrypted Payloads

Warning: This page is still under development and not complete. It will be so until this warning is removed.

2.4 Agent Revocation

Warning: This page is still under development and not complete. It will be so until this warning is removed.

CHAPTER 3

Rest API's

https://docs.readthedocs.io/en/stable/api/v2.html

All Keylime APIs use *REST (Representational State Transfer)*.

3.1 Authentication and authorization

Not yet implemented

3.2 RESTful API for Keylime (v2)

3.2.1 Cloud verifier (CV)

GET /v2/agents/{agent_id:UUID}
Get status of agent *agent_id* from CV

POST /v2/agents/{agent_id:UUID}
Add new agent *instance_id* to CV

Requires JSON Body:

```
{
  "v" : key,
  "ip" : ipaddr,
  "port" : int,
  "operational_state" : int,
  "public_key" : key,
  "tpm_policy" : json,
  "vtpm_policy" : json,
  "metadata" : json,
  "ima_whitelist" : json,
```

(continues on next page)

(continued from previous page)

```
  "accept_tpm_hash_algs": list,
  "accept_tpm_encryption_algs": list,
  "accept_tpm_signing_algs": list,
}

.. http:delete::  /v2/agents/{agent_id:UUID}

   Terminate instance `agent_id`
```

PUT /v2/agents/{agent_id:UUID}/reactivate
Start agent *agent_id*‘ (for an already bootstrapped *agent_id* node)

PUT /v2/agents/{agent_id:UUID}/stop
Stop cv polling on *agent_id*, but don’t delete (for an already started *agent_id*)

3.2.2 Cloud Agent

GET /v2/keys/pubkey
Retrieves agents public key

POST /v2/keys/vkey
Send *v_key* to node

Requires JSON Body:

```
{
  "encrypted_key": key,
}
```

POST /v2/keys/ukey
Send *u_key* to node (with optional payload)

Requires JSON Body:

```
{
  "auth_tag" : hmac,
  "encrypted _key": key,
  "payload": b64, (opt)
}
```

GET /v2/keys/pubkey
Get confirmation of bootstrap key derivation

Requires query parameters:

```
challenge : int
```

GET /v2/quotes/integrity
Get integrity quote from node

Required parameters:

```
nonce : int
mask : bitmask
vmask : bitmask
partial : bool
```

Example:

```
/v2/quotes/integrity?nonce=#&mask=#&vmask=#&partial=#
```

GET /v2/quotes/identity
Get identity quote from node

Required parameters:

```
nonce : int
```

Example:

```
/v2/quotes/identity?nonce=#
```

3.2.3 Cloud verifier (CV)

GET /v2/agents/
Get ordered list of registered agents

GET /v2/agents/{agent_id:UUID}
Get AIK of agent *agent_id*

POST /v2/agents/{agent_id:UUID}
Add agent *agent_id* to registrar

Requires JSON Body:

```
{
  "ek" : key,
  "ekcert" : cert,
  "aik" : key,
  "tpm_version": TPM version,
  "aik_name" : key name, (tpm2)
  "ek_tpm" : TPM-format key (tpm2)
}
```

DELETE /v2/agents/{agent_id:UUID}
Remove agent *agent_id* from registrar

PUT /v2/agents/{agent_id:UUID}/activate
Activate physical agent *agent_id*

Requires JSON Body:

```

```

{ "auth_tag" : hmac,

}

PUT /v2/agents/{agent_id:UUID}/vactivate
Activate virtual (vTPM) agent *agent_id*

Requires JSON Body:

```

```

{ “deepquote” : b64,

}

3.2.4 Tenant WebApp

GET /v2/agents/
Get ordered list of registered agents

GET /v2/agents/{agent_id:UUID}
Get list of registered agents

PUT /v2/agents/{agent_id:UUID}
Start agent *agent_id* (For an already bootstrapped *agent_id* agent)

POST /v2/agents/{agent_id:UUID}
Add agent *agent_id* to registrar

Requires JSON Body:

```
{
  “ip” : ipaddr,
  “keyfile_data” : base64,
  “keyfile_name” : string, (opt)
  “file_data” : base64,
  “file_name” : string, (opt)
  “ca_dir” : string,
  “ca_dir_pw” : string,
  “include_dir_data” : base64,
  “include_dir_name” : string,
}
```

GET /v2/logs/
Get terminal log data

GET /v2/logs/{logType:string}
Get terminal log data for given logType

Optional query parameters:

```
pos : int, (opt)
```

Example:

```
/v2/logs/tenant?pos=#
```

3.3 RESTful API Responses for Keylime (v2)

CHAPTER 4

KeyLime Development

4.1 Contributing

When contributing any keylime repository, please first discuss the change you wish to make via an issue in the relevant repository for your change or email to the keylime mailing list

4.1.1 Pull Request Process

1. Create an issue outlining the fix or feature.
2. Fork the keylime repository to your own github account and clone it locally.
3. Hack on your changes.
4. Update the README.md or documentation with details of changes to any interface, this includes new environment variables, exposed ports, useful file locations, CLI parameters and configuration values.
5. Add and commit your changes with some descriptive text on the nature of the change / feature in your commit message. Also reference the issue raised at [1] as follows: *Fixes #45*. See the following link for more message types
6. Ensure that CI passes, if it fails, fix the failures.
7. Every pull request requires a review from the core keylime team
8. If your pull request consists of more than one commit, please squash your commits as described in see *Squash Commits*.

4.2 Commit Message Guidelines

We follow the commit formatting recommendations found on Chris Beams' How to Write a Git Commit Message article.

Well formed commit messages not only help reviewers understand the nature of the Pull Request, but also assists the release process where commit messages are used to generate release notes.

A good example of a commit message would be as follows:

```
Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72
characters or so. In some contexts, the first line is treated as the
subject of the commit and the rest of the text as the body. The
blank line separating the summary from the body is critical (unless
you omit the body entirely); various tools like `log`, `shortlog`
and `rebase` can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you
are making this change as opposed to how (the code explains that).
Are there side effects or other unintuitive consequences of this
change? Here's the place to explain them.

Further paragraphs come after blank lines.

- Bullet points are okay, too

- Typically a hyphen or asterisk is used for the bullet, preceded
  by a single space, with blank lines in between, but conventions
  vary here

If you use an issue tracker, put references to them at the bottom,
like this:

Resolves: #123
See also: #456, #789
```

Note the *Resolves #123* tag, this references the issue raised and allows us to ensure issues are associated and closed when a pull request is merged.

Please refer to the github help page on message types for a complete list of issue references.

4.3 Squash Commits

Should your pull request consist of more than one commit (perhaps due to a change being requested during the review cycle), please perform a git squash once a reviewer has approved your pull request.

A squash can be performed as follows. Let's say you have the following commits:

```
initial commit
second commit
final commit
```

Run the command below with the number set to the total commits you wish to squash (in our case 3 commits):

```
git rebase -i HEAD~3
```

You default text editor will then open up and you will see the following:

```
pick eb36612 initial commit
pick 9ac8968 second commit
pick a760569 final commit

# Rebase eb1429f..a760569 onto eb1429f (3 commands)
```

We want to rebase on top of our first commit, so we change the other two commits to *squash*:

```
pick eb36612 initial commit
squash 9ac8968 second commit
squash a760569 final commit
```

After this, should you wish to update your commit message to better summarise all of your pull request, run:

```
git commit --amend
```

You will then need to force push (assuming your initial commit(s) were posted to github):

```
git push origin your-branch --force
```

4.4 Docker Development Environment

The following is a guide to mounting your local repository as a Docker volume and performing a test run using a TPM simulator. This will replicate the same test that occurs within the KeyLime CI gate for keylime.

This requires a working installation of Docker. See your distributions guide on how to set that up.

As an example, on Fedora 29:

```
sudo dnf -y install dnf-plugins-core
sudo dnf install docker-ce docker-ce-cli containerd.io
sudo usermod -aG docker $USER
sudo systemctl enable docker
sudo systemctl start docker
```

Note: login and out of your shell, if you want to run docker as *$USER*

Save the following script to your local machine (tip: create an alias to call the script in an easy to remember way):

```
#!/bin/bash

# Your local keylime. (you should likely change this)
REPO="/home/${USER}/keylime."

# keylime images
tpm12image="lukehinds/keylime-ci-tpm12"
tpm12tag="v300"
tpm20image="lukehinds/keylime-ci-tpm20"
tpm20tag="v301"

echo -e "Grabbing latest images"

docker pull ${tpm12image}:${tpm12tag}
docker pull ${tpm20image}:${tpm20tag}

function tpm1 {
```

(continues on next page)

(continued from previous page)

```
    container_id=$(mktemp)
    docker run --detach --privileged \
        -v $REPO:/root/keylime. \
        -it ${tpm12image}:${tpm12tag} >> ${container_id}
    docker exec -u 0 -it --tty "$(cat ${container_id})" \
        /bin/sh -c 'cd /root/keylime./test; chmod +x ./run_tests.sh; ./run_tests.sh -
↪s openssl'
    docker stop "$(cat ${container_id})"
    docker rm "$(cat ${container_id})"
}

function tpm2 {
    container_id=$(mktemp)
    docker run --detach --privileged \
        -v $REPO:/root/keylime. \
        -v /sys/fs/cgroup:/sys/fs/cgroup:ro \
        -it ${tpm20image}:${tpm20tag} >> ${container_id}
    docker exec -u 0 -it --tty "$(cat ${container_id})" \
        /bin/bash /root/keylime./.ci/test_wrapper.sh
    docker stop "$(cat ${container_id})"
    docker rm "$(cat ${container_id})"
}

while true; do
    echo -e ""
    read -p "Do you wish to test against TPM1.2(a) / TPM 2.0(b) or q(quit): " abq
    case $abq in
        [a]* ) tpm1;;
        [b]* ) tpm2;;
        [q]* ) exit;;
        * ) echo "Please answer 1, 2 q(quit)";;
    esac
done
```

CHAPTER 5

Securing Keylime

Warning: This page is still under development and not complete. It will be so until this warning is removed.

5.1 System Hardening

5.2 TLS configuration

5.3 Reporting an issue

Please contact us directly at security@keylime.groups.io for any bug that might impact the security of this project. Do not use a github issue to report any potential security bugs.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

/v2