PyKMIP as vcenter KSM server

Veröffentlicht am **22. September 2018**

There are multiple reasons why somebody would like to have a KSM server.
This article explains how to set up such a server with persistent database storage, so that an encrypted vm survives a complete (vcenter/esxi) reboot.
Although it works, it is certainly not recommended to do it this way in large environments.
But nevertheless, it's a cheap way to get a vtpm.
Remark: After a reboot of vcenter, the trust seems still intact, but does not work anymore! In that case, just remove the KMS cluster-definition and re-add and trust it. After that you should be able to unlock encrypted VMs again.

I did this with an minimal Ubuntu server 18.04.1 64bit install. The service runs as the admin user (sudoer, not root). You might want to change that to even a less privileges user and probably setup iptables or other means to protect the TPM data! It is an sqlite db, so anybody with file-access could steal and read it! This tutorial also uses a self-signed certificate. If you have a PKI, it would certainly be better to use properly signed and managed ssl certs…
You have been warned! 😉

<$username> should get replaced with the actual username.
Commands in green should be executed as user.
Commands in red should get executed as root.
The switch between the two are the sudo and exit commands.

Login as user to the ubuntu machine.

```
sudo -i
apt-get update
apt-get upgrade
mkdir /usr/local/PyKMIP
mkdir /etc/pykmip
mkdir /var/log/pykmip
chown <$username>: -R /usr/local/PyKMIP
chown <$username>: -R /etc/pykmip
chown <$username>: -R /var/log/pykmip
apt -get installpython-dev libffi-dev libssl-dev libsqlite3-dev python-setuptools python-requests
openssl req -x509 -nodes -days 9999 -newkey rsa:2048 -keyout /etc/ssl/private/selfsigned.key -out /etc/ssl/certs/selfsigned.crt
```

Fill out the form…

```
chown <$username>: -R /etc/ssl/private
chown <$username>: /etc/ssl/certs/selfsigned.crt
exit
cd /usr/local
```

If you need to use a proxy, then replace X.X.X.X with the ip of the proxy and PORT with the port your proxy server is available. You might also add "yourproxyusername:yourproxypassword@" directly in front of the ip, if your proxy requires authentication. We will need this one more time later on, so keep them in mind.
If you don't need any proxy, the you can leave the following two commands out.

```
export https_proxy=http://X.X.X.X:PORT
export http_proxy=http://X.X.X.X:PORT

cd /usr/local
git clone https://github.com/OpenKMIP/PyKMIP

sudo -i
```

Again, the Proxy, but this time as root.

```
export https_proxy=http://X.X.X.X:PORT
export http_proxy=http://X.X.X.X:PORT

cd /usr/local/PyKMIP
python setup.py install
exit
nano /etc/pykmip/server.conf
```

Enter these following lines between the — signs (but without them) in the nano editor.
Replace hostname=10.X.X.X with the servers IP.
Quit with ctrl-x followed by y and enter

—
```
[server]
database_path=/etc/pykmip/pykmip.database
hostname=10.X.X.X
port=5696
certificate_path=/etc/ssl/certs/selfsigned.crt
key_path=/etc/ssl/private/selfsigned.key
ca_path=/etc/ssl/certs/selfsigned.crt
auth_suite=TLS1.2
policy_path=/usr/local/PyKMIP/examples/
enable_tls_client_auth=False
tls_cipher_suites=
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
logging_level=DEBUG
```
—

We should now be ready to start the service!

```
cd /usr/local/PyKMIP
python bin/run_server.py
```

Now you should be able to add the host as KSM server in vcenter with the ip and port 5696.

To make the connection complete, you need to press the “Make KMS trust vcenter” button.
Choose “KMS certificate and private key”
Open a new shell to the ubuntu server
```
cat /etc/ssl/certs/selfsigned.crt
```
copy the whole output inclusive the —begin and end — messages and paste it to the first field “KMS Certificate” in vcenter
```
cat /etc/ssl/private/selfsigned.key
```
copy the whole output inclusive the —begin and end — messages and paste it to the second field “KMS Private Key” in vcenter

Press the “Establish Trust” button.

To let the service start on every boot, you can add it to the crontab.

```
crontab -e
```

Add the following line. and save the file.

```
@reboot ( sleep 30s; python /usr/local/PyKMIP/bin/run_server & )
```