安装与配置

本章节介绍如何在控制节点上安装和配置Barbican密钥管理服务。
本章节假定您已安装了一个拥有身份认证服务（Keystone）的OpenStack环境。 为方便起见，此配置将机密存储在本地文件系统上。
安装环境为**Ubuntu**。

安装

安装前准备

在安装密钥管理服务之前，你必需向创建数据库、服务凭证和API端点。

1. 要创建数据库，请完整执行以下步骤：
   1.1 使用**root**用户访问数据库。
   1.2 创建**barbican**数据库：

```
                CREATE DATABASE barbican;
```

   1.3 赋予**barbican**用户访问数据库的权限：

```
                GRANT ALL PRIVILEGES ON barbican.* TO
 'barbican'@'localhost' IDENTIFIED BY 'BARBICAN_DBPASS';
                GRANT ALL PRIVILEGES ON barbican.* TO 'barbican'@'%'
 IDENTIFIED BY 'BARBICAN_DBPASS';
```

使用合适的密码替换**BARBICAN_DBPASS**。

2. source管理员凭证来执行只有管理才能访问的CLI命令：

```
    $ source admin-openrc
```

3. 要创建服务凭证，请完整执行以下命令：
   3.1 创建**barbican**用户：

```
        $ openstack user create --domain default --password-prompt
 barbican
```

   3.2 为**barbican**用户添加**admin**角色：

```
        $ openstack role add --project service --user barbican admin
```

3.3 创建**createor**角色：

```
        $ openstack role create creator
```

3.4 为**barbican**用户添加**creator**角色：

```
        $ openstack role add --project service --user barbican creator
```

3.5 创建barbican服务实例：

```
        $ openstack service create --name barbican --description "Key
Manager" key-manager
```

4. 创建密钥管理服务端点：

```
    $ openstack endpoint create --region RegionOne key-manager public
http://controller:9311
    $ openstack endpoint create --region RegionOne key-manager internal
http://controller:9311
    $ openstack endpoint create --region RegionOne key-manager admin
http://controller:9311
```

安装和配置组件

1. 安装barbican包

```
    # apt-get update
    # apt-get install barbican-api barbican-keystone-listener barbican-
worker
```

2. 按如下操作编辑 **/etc/barbican/barbican.conf**文件：

2.1 在 **[DEFAULT]** 部分中配置数据库访问：

```
        [DEFAULT]
        ...
        sql_connection =
mysql+pymysql://barbican:BARBICAN_DBPASS@controller/barbican
```

使用你的数据库访问密码替换 **BARBICAN_DBPASS** 访问数据库。

2.2 在 **[DEFAULT]** 部分中配置RabbitMQ消息队列访问：

```
[DEFAULT]
...
rabbit://openstack:RABBIT_PASS@controller
```

用你为**OpenStack**选择的在**RabbitMQ**中使用的密码替换**RABBIT_PASS**。

2.3 在 **[keystone_authtoken]**中配置身份服务访问：

```
[keystone_authtoken]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = barbican
password = BARBICAN_PASS
```

用你为**barbican**用户选择的在身份服务中使用的密码替换**BARBICAN_PASS**。

3. 填充密钥管理服务数据库：

初次启动时，密钥管理服务时会自动填充数据库。为防止这样，可以手动执行数据库同步：编辑**/etc//barbican/barbican.conf**文件，将其中**[DEFAULT]**部分中的*db_auto_create*设置为false。

然后执行以下命令进行数据库填充：

```
$ su -s /bin/sh -c "barbican-manage db upgrade" barbican
```

4. Barbican有一个插件框架，允许部署者在许多不同机密存储后端存储机密。通常情况下，Barbican被配置为使用基于文件基础的密钥库存储机密，这种方式不适合在生产环境中使用。

查看支持的插件列表节如何配置它们的细节介绍，详见配置机密存储后端。

安装最后

重启密钥管理服务：

```
# service barbican-keystone-listener restart
# service barbican-worker restart
# service apache2 restart
```

配置机密存储后端

密钥管理服务有一个插件架构，允许部署者使用一个或者多个机密存储来存储机密。机密存储可以是基于软件的，例如单纯的软件加密机制；或者基于硬件驱动，例如硬件安全模块（HSM）。

机密存储不仅提供加密机制，也提供加密机密的存储。

本章节将比较目前所有可用的插件以及在选择时应该考虑的安全取舍。

简单加密插件

此后端插件仅提供软件加密。加密后的机密存储在barbican数据库中。

在 **/etc/barbican/barbican.conf**中，此插件被配置为默认。

此插件使用文本格式存储在 **/etc/barbican/barbican.conf**文件中的单个对称密钥加解密所有机密。

安全性	**主密钥（KEK）存储在配置文件中**
是否成熟	每个补丁上进行测试
易于使用	方便部署；密钥旋转是破坏性的，所有的机密都需要重新加密；可以在SQL DB扩展中存储
可扩展性	故障转移/HA很简单，只需执行barbican-api实例即可
成本	高性能，软件加密快；免费

警告

> 此插件的KEK以纯文本的形式存储在配置文件中，该文件可以存在任何运行barbican-api或barbican-worker服务的节点中。应该特别注意防止对这些节点未经授权的访问。使用此插件时，KEK是唯一保护数据库中存储的机密的东西。

此插件在 **/etc/barbican/barbican.conf**文件中的配置如下：

```
# ================= Secret Store Plugin ===================
[secretstore]
..
enabled_secretstore_plugins = store_crypto

# ================= Crypto plugin ===================
[crypto]
..
enabled_crypto_plugins = simple_crypto

[simple_crypto_plugin]
# the kek should be a 32-byte value which is base64 encoded
kek = 'YWJjZGVmZ2hpamtsbW5vcHFyc3R1dnd4eXoxMjM0NTY='
```

PKCS#11插件

此插件使用PKCS#11协议与硬件安全模块（HSM）连接。
通过项目特定的密钥加密密钥（KEK）对机密进行加密（检索时对解密），KEK驻留在HSM中。
此插件的配置在 **/etc/barbican/barbican.conf**中。下面给出几个不同HSM的设置：

Safenet

PKCS#11插件配置如下：

```
# ================= Secret Store Plugin ===================
[secretstore]
..
enabled_secretstore_plugins = store_crypto

[p11_crypto_plugin]
# Path to vendor PKCS11 library
library_path = '/usr/lib/libCryptoki2_64.so'

# Password to login to PKCS11 session
login = 'mypassword'

# Label to identify master KEK in the HSM (must not be the same as HMAC
label)
mkek_label = 'an_mkek'

# Length in bytes of master KEK
mkek_length = 32

# Label to identify HMAC key in the HSM (must not be the same as MKEK
label)
hmac_label = 'my_hmac_label'

# HSM Slot id (Should correspond to a configured PKCS11 slot). Default: 1
# slot_id = 1

# Enable Read/Write session with the HSM?
# rw_session = True

# Length of Project KEKs to create
# pkek_length = 32

# How long to cache unwrapped Project KEKs
# pkek_cache_ttl = 900

# Max number of items in pkek cache
# pkek_cache_limit = 100
```

Thales

Thales NetHSM插件的配置如下：

```
# ================= Secret Store Plugin ===================
[secretstore]
..
enabled_secretstore_plugins = store_crypto

[p11_crypto_plugin]
# Path to vendor PKCS11 library
library_path = '/opt/nfast/toolkits/pkcs11/libcknfast.so'

# Password to login to PKCS11 session
login = 'XXX'

# Label to identify master KEK in the HSM (must not be the same as HMAC
label)
mkek_label = 'thales_mkek_0'

# Length in bytes of master KEK
mkek_length = 32

# Label to identify HMAC key in the HSM (must not be the same as MKEK
label)
hmac_label = 'thales_hmac_0'

# HSM Slot id (Should correspond to a configured PKCS11 slot). Default: 1
# slot_id = 1

# Enable Read/Write session with the HSM?
# rw_session = True

# Length of Project KEKs to create
# pkek_length = 32

# How long to cache unwrapped Project KEKs
# pkek_cache_ttl = 900

# Max number of items in pkek cache
# pkek_cache_limit = 100

# Secret encryption mechanism (string value)
# Deprecated group/name - [p11_crypto_plugin]/algorithm
encryption_mechanism = CKM_AES_CBC

# HMAC Key Type (string value)
hmac_key_type=CKK_SHA256_HMAC

# HMAC Key Generation Mechanism (string value)
hmac_keygen_mechanism = CKM_NC_SHA256_HMAC_KEY_GEN

# Generate IVs for CKM_AES_GCM mechanism. (boolean value)
# Deprecated group/name - [p11_crypto_plugin]/generate_iv
aes_gcm_generate_iv=True
```

```
# Always set CKA_SENSITIVE=CK_TRUE including
# CKA_EXTRACTABLE=CK_TRUE keys.
# default true
always_set_cka_sensitive=false
```

HMAC和MKEK密钥生成方式如下：

```
barbican-manage hsm gen_hmac --library-path
/opt/nfast/toolkits/pkcs11/libcknfast.so --passphrase XXX --slot-id 1 --
label thales_hmac_0 --key-type CKK_SHA256_HMAC --mechanism
CKM_NC_SHA256_HMAC_KEY_GEN
```

```
barbican-manage hsm gen_mkek --library-path
/opt/nfast/toolkits/pkcs11/libcknfast.so --passphrase XXX --slot-id 1 --
label thales_mkek_0
```

ATOS Bull

ATOS Bull HSM插件配置如下：

```
# ================= Secret Store Plugin ===================
[secretstore]
..
enabled_secretstore_plugins = store_crypto

[p11_crypto_plugin]
# Path to vendor PKCS11 library
library_path = '/usr/lib64/libnethsm.so'

# Password to login to PKCS11 session
login = 'XXX'

# Label to identify master KEK in the HSM (must not be the same as HMAC
label)
mkek_label = 'atos_mkek_0'

# Length in bytes of master KEK
mkek_length = 32

# Label to identify HMAC key in the HSM (must not be the same as MKEK
label)
hmac_label = 'atos_hmac_0'

# HSM Slot id (Should correspond to a configured PKCS11 slot). Default: 1
# slot_id = 1
```

```
# Enable Read/Write session with the HSM?
# rw_session = True

# Length of Project KEKs to create
# pkek_length = 32

# How long to cache unwrapped Project KEKs
# pkek_cache_ttl = 900

# Max number of items in pkek cache
# pkek_cache_limit = 100

# Secret encryption mechanism (string value)
# Deprecated group/name - [p11_crypto_plugin]/algorithm
encryption_mechanism = CKM_AES_CBC

# HMAC Key Type (string value)
hmac_key_type = CKK_GENERIC_SECRET

# HMAC Key Generation Mechanism (string value)
hmac_keygen_mechanism = CKM_GENERIC_SECRET_KEY_GEN

# Always set CKA_SENSITIVE=CK_TRUE including
# CKA_EXTRACTABLE=CK_TRUE keys.
# default true
always_set_cka_sensitive=false
```

HMAC和MKEK密钥生成方式如下：

```
barbican-manage hsm gen_hmac --library-path /usr/lib64/libnethsm.so --
passphrase XXX --slot-id 1 --label atos_hmac_0 --key-type
CKK_GENERIC_SECRET --mechanism  CKM_GENERIC_SECRET_KEY_GEN
```

```
barbican-manage hsm gen_mkek --library-path /usr/lib64/libnethsm.so --
passphrase XXX --slot-id 1 --label atos_mkek_0
```

KMIP 插件

此机密存储插件用于与KMIP设备进行通信。机密直接安全存储在KMIP设备中，而不是barbican数据库中。
barbican数据库维护机密数据的本地应用，以便之后的检索。
此插件可以配置为使用用户名/密码的方式与KMIP设备认证，也可使用客户端证书认证。
此插件在 **/etc/barbican/barbican.conf**中的配置如下：

```
[secretstore]
..
enabled_secretstore_plugins = kmip_crypto
```

```
[kmip_plugin]
username = 'admin'
password = 'password'
host = localhost
port = 5696
keyfile = '/path/to/certs/cert.key'
certfile = '/path/to/certs/cert.crt'
ca_certs = '/path/to/certs/LocalCA.crt'
```

Dogtag 插件

Dogtag是一个与红毛证书系统对应的上游项目红帽证书系统是一个功能强大的全功能PKI解决方案，提供包括证书管理（CA）和被用来安全存储机密的密钥恢复权限（KRA）。
KRA将机密作为加密的二进制数据存储在其内部数据库中，其主加密密钥可以存储在基于软件的NSS安全数据库，也可存储在HSM中。
注意，基于软件的NSS数据库为那些不需要或无法负担HSM的部署提供了安全选项，这是目前唯一提供此选项的插件。
KRA使用PKCS#11与HSM通信。有关经过认证的HSM列表，参见发行说明。Dogtag和KRA符合所有相关的通用标准和FIPS规范。
KRA是FreeIPA的一个组成部分。因此，可以通过FreeIPA服务器配置插件。
该插件使用可信的KRA代理的客户端证书与KRA通信。该证书存储在NSS数据库以及PEM文件中。
此插件在 **/etc/barbican/barbican.conf**文件中的配置如下：

```
[secretstore]
..
enabled_secretstore_plugins = dogtag_crypto

[dogtag_plugin]
pem_path = '/etc/barbican/kra_admin_cert.pem'
dogtag_host = localhost
dogtag_port = 8443
nss_db_path = '/etc/barbican/alias'
nss_password = 'password123'
```