Red Hat OpenStack Platform 13

OpenStack Key Manager를 사용하여 시크릿 관리

OpenStack 배포에 OpenStack Key Manager(Barbican)를 통합하는 방법

Last Updated: 2023-05-25

Red Hat OpenStack Platform 13 OpenStack Key Manager를 사용하여 시크릿 관리

OpenStack 배포에 OpenStack Key Manager(Barbican)를 통합하는 방법

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

OpenStack 배포에 OpenStack Key Manager(Barbican)를 통합하는 방법

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체 3

1장. 개요 4

2장. 백엔드 선택 5
2.1. 백엔드 간 마이그레이션 5

3장. BARBICAN 설치 6
3.1. OVERCLOUD의 CREATOR 역할에 사용자 추가 7
3.2. 정책 이해 9

4장. 바비칸에서 시크릿 관리 11
4.1. 보안 나열 11
4.2. 새 보안 추가 11
4.3. 보안 업데이트 11
4.4. 보안 삭제 11
4.5. 대칭 키 생성 12
4.6. 키 백업 및 복원BACKUP AND RESTORE KEYS 13

5장. CINDER 볼륨 암호화 17
5.1. 기존 볼륨 키를 BARBICAN으로 마이그레이션 19

6장. AT-REST SWIFT 오브젝트 암호화 23
6.1. SWIFT에 대해 유휴 상태의 암호화 활성화 23

7장. GLANCE 이미지 검증 24
7.1. GLANCE 이미지 검증 활성화 24
7.2. 이미지 검증 24

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

1장. 개요

OpenStack Key Manager(barbican)는 Red Hat OpenStack Platform의 시크릿 관리자입니다. barbican API 및 명령줄을 사용하여 OpenStack 서비스에서 사용하는 인증서, 키 및 암호를 중앙에서 관리할 수 있습니다. Barbican 현재 이 가이드에서 설명하는 다음과 같은 사용 사례를 지원합니다.

- 대칭 **암호화 키** - 블록 스토리지(cinder) 볼륨 암호화, 임시 디스크 암호화 및 Object Storage(swift) 암호화에 사용됩니다.
- **비대칭 키 및 인증서** - Glance 이미지 서명 및 확인에 사용됩니다.

이번 릴리스에서는 Barbican에서 Block Storage(cinder) 및 Compute(nova) 구성 요소와의 통합을 제공할 수 있습니다.

2장. 백엔드 선택

시크릿(예: 인증서, API 키 및 암호)은 barbican 데이터베이스에 암호화 Blob으로 저장하거나 보안 스토리지 시스템에 직접 저장할 수 있습니다.

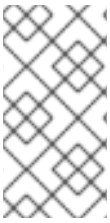
시크릿을 Barbican 데이터베이스에 암호화된 Blob으로 저장하려면 다음 옵션을 사용할 수 있습니다.

- **단순 암호화 플러그인** - 간단한 암호화 플러그인은 기본적으로 활성화되어 있으며 단일 대칭 키를 사용하여 보안 Blob을 암호화합니다. 이 키는 **barbican.conf** 파일의 일반 텍스트로 저장됩니다.

참고

간단한 암호화 플러그인은 현재 Red Hat에서 지원하는 유일한 플러그인입니다.

- **PKCS#11 암호화 플러그인** - PKCS#11 암호화 플러그인은 barbican 데이터베이스에 저장된 프로젝트별 키 암호화 키(KEK)를 사용하여 시크릿을 암호화합니다. 이러한 프로젝트별 KEK는 하드웨어 보안 모듈(HSM)에 저장되는 마스터 KEK에 의해 암호화됩니다. 모든 암호화 및 암호 해독 작업은 프로세스 내 메모리가 아닌 HSM에서 수행됩니다.All encryption and decryption operations take place in the HSM, rather than in-process memory. PKCS#11 플러그인은 PKCS#11 프로토콜을 통해 HSM과 통신합니다. 암호화는 안전한 하드웨어에서 수행되고 프로젝트마다 다른 KEK가 사용되므로 이 옵션은 간단한 암호화 플러그인보다 더 안전합니다.



참고

HA(고가용성) 옵션과 관련하여 barbican 서비스는 Apache 내에서 실행되며 고가용성을 위해 HAProxy를 사용하도록 director에서 구성합니다. 백엔드 계층의 HA 옵션은 사용 중인 백엔드에 따라 달라집니다. 예를 들어 간단한 crypto의 경우 모든 barbican 인스턴스에 구성 파일에 동일한 암호화 키가 있어 간단한 HA 구성이 생성됩니다.

2.1. 백엔드 간 마이그레이션

Barbican을 사용하면 프로젝트에 대한 다른 백엔드를 정의할 수 있습니다. 프로젝트에 대한 매핑이 없는 경우 시크릿은 글로벌 기본 백엔드에 저장됩니다. 즉, 여러 백엔드를 구성할 수 있지만, 하나 이상의 글로벌 백엔드가 정의되어 있어야 합니다. 다른 백엔드에 대해 제공되는 heat 템플릿에는 각 백엔드를 기본값으로 설정하는 매개변수가 포함되어 있습니다.

특정 백엔드에 시크릿을 저장한 다음 새 백엔드로 마이그레이션하도록 결정하는 경우 새 백엔드를 글로벌 기본값(또는 프로젝트별 백엔드)으로 활성화하는 동안 이전 백엔드를 계속 사용할 수 있습니다. 결과적으로 이전 시크릿은 이전 백엔드를 통해 계속 사용할 수 있습니다.

3장. BARBICAN 설치

Barbican은 Red Hat OpenStack Platform에서 기본적으로 활성화되어 있지 않습니다. 다음 절차에서는 기존 OpenStack 배포에 barbican을 배포하는 방법을 설명합니다. Barbican은 컨테이너화된 서비스로 실행되므로 이 절차에서는 새 컨테이너 이미지를 준비하고 업로드하는 방법도 설명합니다.

참고

이 절차에서는 **simple_crypto** 백엔드를 사용하도록 barbican을 구성합니다. **PKCS11** 및 DogTag와 같은 추가 백엔드는 사용할 수 있지만 이 릴리스에서는 지원되지 않습니다.

1. 언더클라우드 노드에서 barbican의 환경 파일을 생성합니다. 그러면 director에 barbican을 설치합니다(*openstack overcloud deploy [...]* 에 포함된 경우)

```
$ cat /home/stack/configure-barbican.yaml
 parameter_defaults:
   BarbicanSimpleCryptoGlobalDefault: true
```

   - **BarbicanSimpleCryptoGlobalDefault** - 이 플러그인을 글로벌 기본 플러그인으로 설정합니다.
   - 추가 옵션도 구성할 수 있습니다.
     - **BarbicanPassword** - barbican 서비스 계정의 암호를 설정합니다.
     - **BarbicanWorkers** - **barbican::wsgi::apache** 의 작업자 수를 설정합니다. 기본적으로 **'%{::processorcount}'** 를 사용합니다.
     - **BarbicanDebug** - 디버깅을 활성화합니다.
     - **BarbicanPolicies** - barbican에 대해 구성할 정책을 정의합니다. 예를 들어 **{ barbican-context_is_admin: { key: context_is_admin, value: 'role:admin' } }** 와 같은 해시 값을 사용합니다. 그러면 이 항목이 **/etc/barbican/policy.json** 에 추가됩니다. 정책은 이후 섹션에서 자세히 설명합니다.
     - **BarbicanSimpleCryptoKek** - 지정된 항목이 없는 경우 director에서 키 암호화 키(KEK)가 생성됩니다.

2. 이 단계에서는 barbican을 위해 새 컨테이너 이미지를 준비합니다. **configure-barbican.yaml** 및 모든 관련 템플릿 파일을 포함해야 합니다. 배포에 맞게 다음 예제를 변경합니다.

```
$ openstack overcloud container image prepare \
 --namespace example.lab.local:5000/rhosp13 \
 --tag 2018-06-06.1 \
 --push-destination 192.168.100.1:8787 \
 --output-images-file ~/container-images-with-barbican.yaml \
 -e /home/stack/virt/config_lvm.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
 -e /home/stack/virt/network/network-environment.yaml \
 -e /home/stack/virt/hostnames.yml \
 -e /home/stack/virt/nodes_data.yaml \
 -e /home/stack/virt/extra_templates.yaml \
 -e /home/stack/virt/docker-images.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml \
```

```
 -e /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-simple-crypto.yaml \
 -e /home/stack/configure-barbican.yaml
```

3. 새 컨테이너 이미지를 언더클라우드 레지스트리에 업로드합니다.

```
$ openstack overcloud container image upload --debug --config-file container-images-with-barbican.yaml
```

4. 새 환경 파일을 준비합니다.

```
$ openstack overcloud container image prepare \
  --tag 2018-06-06.1 \
  --namespace 192.168.100.1:8787/rhosp13 \
  --output-env-file ~/container-parameters-with-barbican.yaml \
  -e /home/stack/virt/config_lvm.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /home/stack/virt/network/network-environment.yaml \
  -e /home/stack/virt/hostnames.yml \
  -e /home/stack/virt/nodes_data.yaml \
  -e /home/stack/virt/extra_templates.yaml \
  -e /home/stack/virt/docker-images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-simple-crypto.yaml \
  -e /home/stack/configure-barbican.yaml
```

5. 이러한 변경 사항을 배포에 적용하려면 오버클라우드를 업데이트하고 이전 *openstack overcloud deploy [...]* 에서 사용한 모든 heat 템플릿 파일을 지정합니다. 예를 들면 다음과 같습니다.

```
$ openstack overcloud deploy \
    --timeout 100 \
    --templates /usr/share/openstack-tripleo-heat-templates \
    --stack overcloud \
    --libvirt-type kvm \
    --ntp-server clock.redhat.com \
    -e /home/stack/virt/config_lvm.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
    -e /home/stack/virt/network/network-environment.yaml \
    -e /home/stack/virt/hostnames.yml \
    -e /home/stack/virt/nodes_data.yaml \
    -e /home/stack/virt/extra_templates.yaml \
    -e /home/stack/container-parameters-with-barbican.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-simple-crypto.yaml \
    -e /home/stack/configure-barbican.yaml \
    --log-file overcloud_deployment_38.log
```

3.1. OVERCLOUD의 CREATOR 역할에 사용자 추가

사용자는 barbican 시크릿을 생성하고 편집하려면 **creator** 역할의 멤버여야 하며, barbican에 자신의 시크릿을 저장하는 암호화된 볼륨을 생성해야 합니다.

1. **creator** 역할의 **id** 를 검색합니다.

```
openstack role show creator
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 4e9c560c6f104608948450fbf316f9d7 |
| name      | creator                          |
+-----------+----------------------------------+
```

참고

OpenStack Key Manager(barbican)가 설치되어 있지 않으면 **creator** 역할이 표시되지 않습니다.

2. 사용자를 **작성자 역할에 할당하고** 관련 프로젝트를 지정합니다. 이 예제에서는 **project_a** 프로젝트에서 **user1** 이라는 사용자가 **creator** 역할에 추가됩니다.

```
openstack role add --user user1 --project project_a 4e9c560c6f104608948450fbf316f9d7
```

3.1.1. 테스트 barbican 기능

이 섹션에서는 바비카가 제대로 작동하는지 테스트하는 방법에 대해 설명합니다.

1. 테스트 시크릿을 생성합니다. 예를 들면 다음과 같습니다.

```
$ openstack secret store --name testSecret --payload 'TestPayload'
+---------------+------------------------------------------------------------------------------------+
| Field         | Value                                                                              |
+---------------+------------------------------------------------------------------------------------+
| Secret href   | https://192.168.123.163/key-manager/v1/secrets/4cc5ffe0-eea2-449d-9e64-
b664d574be53 |
| Name          | testSecret                                                                         |
| Created       | None                                                                               |
| Status        | None                                                                               |
| Content types | None                                                                               |
| Algorithm     | aes                                                                                |
| Bit length    | 256                                                                                |
| Secret type   | opaque                                                                             |
| Mode          | cbc                                                                                |
| Expiration    | None                                                                               |
+---------------+------------------------------------------------------------------------------------+
```

2. 방금 생성한 보안에 대한 페이로드를 검색합니다.

```
openstack secret get https://192.168.123.163/key-manager/v1/secrets/4cc5ffe0-eea2-449d-
9e64-b664d574be53 --payload
+---------+------------+
| Field   | Value      |
+---------+------------+
| Payload | TestPayload |
+---------+------------+
```

3.2. 정책 이해

Barbican은 정책을 사용하여 키 추가 또는 삭제와 같은 시크릿에 대해 작업을 수행할 수 있는 사용자를 결정합니다. 이러한 제어를 구현하기 위해 이전에 만든 keystone 프로젝트 역할(예: 이전에 만든 **생성자**)은 Barbican 내부 권한에 매핑됩니다. 결과적으로 해당 프로젝트 역할에 할당된 사용자는 해당 barbican 권한을 수신합니다.

3.2.1. 기본 정책 보기

기본 정책은 코드에 정의되어 있으며 일반적으로 수정할 필요가 없습니다. **barbican** 소스 코드에서 기본 정책을 생성하여 이를 생성할 수 있습니다.

1. 추가 구성 요소가 다운로드 및 설치될 수 있으므로 프로덕션 환경 외 시스템에서 다음 단계를 수행합니다. 이 예제에서는 **queens** 분기로 전환하므로 다른 버전을 사용하는 경우 이 값을 적용해야 합니다.

```
git clone https://github.com/openstack/barbican
cd /home/stack/barbican
git checkout origin/stable/queens
tox -e genpolicy
```

   이렇게 하면 기본 설정 **etc/barbican/policy.yaml.sample** 이 포함된 하위 디렉터리에 정책 파일이 생성됩니다. 이 경로는 시스템의 **/etc** 디렉터리가 아닌 리포지토리의 하위 디렉터리를 참조합니다. 이 파일의 내용은 다음 단계에서 설명합니다.

2. 생성한 **policy.yaml.sample** 파일은 barbican에서 사용하는 정책을 설명합니다. 이 정책은 사용자가 시크릿 및 시크릿 메타데이터와 상호 작용하는 방법을 정의하는 네 가지 다른 역할로 구현됩니다. 사용자는 특정 역할에 할당함으로써 이러한 권한을 받습니다.

   - **admin** - 시크릿을 삭제, 생성/편집할 수 있습니다.
   - **Creator** - 시크릿을 만들고 읽을 수 있습니다. 시크릿을 삭제할 수 없습니다.
   - **관찰자** - 데이터를 읽을 수 있습니다.
   - **audit** - 메타데이터만 읽을 수 있습니다. 시크릿을 읽을 수 없습니다.
     예를 들어 다음 항목은 각 프로젝트의 **admin**,**observer**, **creator** keystone 역할을 나열합니다. 오른쪽에는 **role:admin**,**role:observer**, **role:creator** 권한이 할당됩니다.

```
#
#"admin": "role:admin"

#
#"observer": "role:observer"

#
#"creator": "role:creator"
```

     이러한 역할은 Barbican으로 그룹화할 수도 있습니다. 예를 들어 **admin_or_creator** 를 지정하는 규칙은 **rule:admin** 또는 **rule:creator** 의 멤버에 적용할 수 있습니다.

3. 파일에 추가로 다운된 **secret:put** 및 **secret:delete** 작업. 오른쪽에는 이러한 작업을 실행할 수 있는 권한이 있는 역할을 확인합니다. 다음 예제에서 **secret:delete** 는 **admin** 및 **creator** 역할 멤버만 시크릿 항목을 삭제할 수 있음을 의미합니다. 또한 규칙에 따라 해당 프로젝트의 **admin** 또는

creator 역할의 사용자가 해당 프로젝트의 시크릿을 삭제할 수 있다고 명시되어 있습니다. 프로젝트 일치는 **secret_project_match** 규칙에 의해 정의되며 정책에도 정의됩니다.

```
secret:delete": "rule:admin_or_creator and rule:secret_project_match"
```

4장. 바비칸에서 시크릿 관리

4.1. 보안 나열

보안은 href 값으로 표시된 해당 URI로 식별됩니다. 이 예에서는 이전 단계에서 생성한 보안을 보여줍니다.

```
$ openstack secret list
+------------------------------------------------------------------------------+------+-------------------------
+--------+-----------------------------------------+----------+-----------+------------+------+------------+
| Secret href                                                                  | Name | Created                 | Status |
Content types                          | Algorithm | Bit length | Secret type | Mode | Expiration |
+------------------------------------------------------------------------------+------+-------------------------
+--------+-----------------------------------------+----------+-----------+------------+------+------------+
| https://192.168.123.169:9311/v1/secrets/24845e6d-64a5-4071-ba99-0fdd1046172e | None | 2018-
01-22T02:23:15+00:00 | ACTIVE | {u'default': u'application/octet-stream'} | aes       |       256 |
symmetric   | None | None       |
+------------------------------------------------------------------------------+------+-------------------------
+--------+-----------------------------------------+----------+-----------+------------+------+------------+
```

4.2. 새 보안 추가

테스트 시크릿을 생성합니다. 예를 들면 다음과 같습니다.

```
$ openstack secret store --name testSecret --payload 'TestPayload'
+---------------+------------------------------------------------------------------------------+
| Field         | Value                                                                        |
+---------------+------------------------------------------------------------------------------+
| Secret href   | https://192.168.123.163:9311/v1/secrets/ecc7b2a4-f0b0-47ba-b451-0f7d42bc1746 |
| Name          | testSecret                                                                   |
| Created       | None                                                                         |
| Status        | None                                                                         |
| Content types | None                                                                         |
| Algorithm     | aes                                                                          |
| Bit length    | 256                                                                          |
| Secret type   | opaque                                                                       |
| Mode          | cbc                                                                          |
| Expiration    | None                                                                         |
+---------------+------------------------------------------------------------------------------+
```

4.3. 보안 업데이트

시크릿을 삭제하는 것 이외의 시크릿의 페이로드를 변경할 수 없지만, 처음 페이로드를 지정하지 않고 시크릿을 생성한 경우 나중에 **업데이트** 함수를 사용하여 페이로드를 추가할 수 있습니다. 예를 들면 다음과 같습니다.

```
$ openstack secret update https://192.168.123.163:9311/v1/secrets/ca34a264-fd09-44a1-8856-
c6e7116c3b16 'TestPayload-updated'
$
```

4.4. 보안 삭제

URI를 지정하여 시크릿을 삭제할 수 있습니다. 예를 들면 다음과 같습니다.

```
$ openstack secret delete https://192.168.123.163:9311/v1/secrets/ecc7b2a4-f0b0-47ba-b451-0f7d42bc1746
$
```

4.5. 대칭 키 생성

대칭 키는 nova 디스크 암호화 및 swift 오브젝트 암호화와 같은 특정 작업에 적합합니다.

1. **order create** 를 사용하여 256비트 키를 생성하고 barbican에 저장합니다. 예를 들면 다음과 같습니다.

```
$ openstack secret order create --name swift_key --algorithm aes --mode ctr --bit-length 256 --payload-content-type=application/octet-stream key
+----------------+-------------------------------------------------------------------------------+
| Field          | Value                                                                         |
+----------------+-------------------------------------------------------------------------------+
| Order href     | https://192.168.123.173:9311/v1/orders/043383fe-d504-42cf-a9b1-bc328d0b4832 |
| Type           | Key                                                                           |
| Container href | N/A                                                                           |
| Secret href    | None                                                                          |
| Created        | None                                                                          |
| Status         | None                                                                          |
| Error code     | None                                                                          |
| Error message  | None                                                                          |
+----------------+-------------------------------------------------------------------------------+
```

   - **--mode** - **ctr** 또는 cbc 와 같은 특정 모드를 사용하도록 생성 키를 구성할 수 **있습니다**. 자세한 내용은 *NIST SP 800-38A* 를 참조하십시오.

2. 여기에 **Secret href** 값으로 표시된 생성된 키의 위치를 식별하는 순서의 세부 정보를 확인합니다.

```
$ openstack secret order get https://192.168.123.173:9311/v1/orders/043383fe-d504-42cf-a9b1-bc328d0b4832
+----------------+-------------------------------------------------------------------------------+
| Field          | Value                                                                         |
+----------------+-------------------------------------------------------------------------------+
| Order href     | https://192.168.123.173:9311/v1/orders/043383fe-d504-42cf-a9b1-bc328d0b4832 |
| Type           | Key                                                                           |
| Container href | N/A                                                                           |
| Secret href    | https://192.168.123.173:9311/v1/secrets/efcfec49-b9a3-4425-a9b6-5ba69cb18719 |
| Created        | 2018-01-24T04:24:33+00:00                                                     |
| Status         | ACTIVE                                                                        |
| Error code     | None                                                                          |
| Error message  | None                                                                          |
+----------------+-------------------------------------------------------------------------------+
```

3. 시크릿 세부 정보를 검색합니다.

```
$ openstack secret get https://192.168.123.173:9311/v1/secrets/efcfec49-b9a3-4425-a9b6-
```

```
5ba69cb18719
+---------------+-------------------------------------------------------------------------------+
| Field         | Value                                                                         |
+---------------+-------------------------------------------------------------------------------+
| Secret href   | https://192.168.123.173:9311/v1/secrets/efcfec49-b9a3-4425-a9b6-
5ba69cb18719 |
| Name          | swift_key                                                                     |
| Created       | 2018-01-24T04:24:33+00:00                                                     |
| Status        | ACTIVE                                                                        |
| Content types | {u'default': u'application/octet-stream'}                                     |
| Algorithm     | aes                                                                           |
| Bit length    | 256                                                                           |
| Secret type   | symmetric                                                                     |
| Mode          | ctr                                                                           |
| Expiration    | None                                                                          |
+---------------+-------------------------------------------------------------------------------+
```

4.6. 키 백업 및 복원BACKUP AND RESTORE KEYS

암호화 키 백업 및 복원 프로세스는 백엔드 유형에 따라 달라집니다.

4.6.1. 간단한 암호화 백엔드를 백업하고 복원하십시오.

간단한 암호화 백엔드를 위해 두 개의 개별 구성 요소(KEK와 데이터베이스)를 백업해야 합니다. 백업 및 복원 프로세스를 정기적으로 테스트하는 것이 좋습니다.

4.6.1.1. 백업 및 KEK 복원

간단한 암호화 백엔드의 경우 마스터 KEK가 포함된 **barbican.conf** 파일을 백업해야 합니다. 이 파일은 보안 강화 위치에 백업해야 합니다. 실제 데이터는 다음 섹션에 설명된 암호화된 상태로 Barbican 데이터베이스에 저장됩니다.

- 백업에서 키를 복원하려면 기존 **barbican.conf** 에 복원된 **barbican.conf** 를 복사해야 합니다.

4.6.1.2. 백엔드 데이터베이스 백업 및 복원

이 절차에서는 간단한 암호화 백엔드에 대한 barbican 데이터베이스를 백업하고 복원하는 방법을 설명합니다. 이를 설명하기 위해 키를 생성하고 barbican에 시크릿을 업로드합니다. 그런 다음 barbican 데이터베이스를 백업하고 생성한 시크릿을 삭제합니다. 그런 다음 데이터베이스를 복원하고 이전에 만든 보안이 복구되었는지 확인합니다.

참고

또한 KEK를 백업했는지도 중요한 요구 사항이므로 백업해야 합니다. 이는 이전 섹션에서 설명합니다.

4.6.1.2.1. 테스트 시크릿 생성

1. 오버클라우드에서 **order create** 를 사용하여 새 256비트 키를 생성하고 barbican에 저장합니다. 예를 들면 다음과 같습니다.

```
(overcloud) [stack@undercloud-0 ~]$ openstack secret order create --name swift_key --
algorithm aes --mode ctr --bit-length 256 --payload-content-type=application/octet-stream key
+---------------+-------------------------------------------------------------------+
```

```
| Field          | Value                                                                |
+----------------+----------------------------------------------------------------------+
| Order href     | http://10.0.0.104:9311/v1/orders/2a11584d-851c-4bc2-83b7-35d04d3bae86 |
| Type           | Key                                                                  |
| Container href | N/A                                                                  |
| Secret href    | None                                                                 |
| Created        | None                                                                 |
| Status         | None                                                                 |
| Error code     | None                                                                 |
| Error message  | None                                                                 |
+----------------+----------------------------------------------------------------------+
```

2. 테스트 보안을 생성합니다.

```
(overcloud) [stack@undercloud-0 ~]$ openstack secret store --name testSecret --payload
'TestPayload'
+---------------+----------------------------------------------------------------------+
| Field         | Value                                                                |
+---------------+----------------------------------------------------------------------+
| Secret href   | http://10.0.0.104:9311/v1/secrets/93f62cfd-e008-401f-be74-bf057c88b04a |
| Name          | testSecret                                                           |
| Created       | None                                                                 |
| Status        | None                                                                 |
| Content types | None                                                                 |
| Algorithm     | aes                                                                  |
| Bit length    | 256                                                                  |
| Secret type   | opaque                                                               |
| Mode          | cbc                                                                  |
| Expiration    | None                                                                 |
+---------------+----------------------------------------------------------------------+
```

3. 보안이 생성되었는지 확인합니다.

```
(overcloud) [stack@undercloud-0 ~]$ openstack secret list
+------------------------------------------------------------------------+-----------+-----------------------
---+--------+-----------------------------------------+----------+-----------+------------+------+------
------+
| Secret href                                                            | Name      | Created               | Status |
Content types                          | Algorithm | Bit length | Secret type | Mode | Expiration |
+------------------------------------------------------------------------+-----------+-----------------------
---+--------+-----------------------------------------+----------+-----------+------------+------+------
------+
| http://10.0.0.104:9311/v1/secrets/93f62cfd-e008-401f-be74-bf057c88b04a | testSecret |
2018-06-19T18:25:25+00:00 | ACTIVE | {u'default': u'text/plain'}              | aes       |        256 |
opaque      | cbc  | None       |
| http://10.0.0.104:9311/v1/secrets/f664b5cf-5221-47e5-9887-608972a5fefb | swift_key  |
2018-06-19T18:24:40+00:00 | ACTIVE | {u'default': u'application/octet-stream'} | aes       |
256 | symmetric   | ctr  | None       |
+------------------------------------------------------------------------+-----------+-----------------------
---+--------+-----------------------------------------+----------+-----------+------------+------+------
------+
```

4.6.1.2.2. barbican 데이터베이스 백업

controller-0 노드에 로그인한 동안 다음 단계를 실행합니다.

참고

barbican 사용자만 *barbican* 데이터베이스에 액세스할 수 있습니다. 따라서 barbican 사용자 암호는 데이터베이스를 백업하거나 복원하는 데 필요합니다.

1. *barbican* 사용자 암호 검색. 예를 들면 다음과 같습니다.

```
[heat-admin@controller-0 ~]$ sudo grep -r "barbican::db::mysql::password"
/etc/puppet/hieradata
/etc/puppet/hieradata/service_configs.json:    "barbican::db::mysql::password":
"seDJRsMNRrBdFryCmNUEFPPev",
```

2. *barbican* 데이터베이스를 백업합니다.

```
[heat-admin@controller-0 ~]$ mysqldump -u barbican -p"seDJRsMNRrBdFryCmNUEFPPev"
barbican > barbican_db_backup.sql
```

3. 데이터베이스 백업은 */home/heat-admin*에 저장됩니다.

```
[heat-admin@controller-0 ~]$ ll
total 36
-rw-rw-r--. 1 heat-admin heat-admin 36715 Jun 19 18:31 barbican_db_backup.sql
```

4.6.1.2.3. 테스트 시크릿 삭제

1. 오버클라우드에서 이전에 생성한 시크릿을 삭제하고 더 이상 존재하지 않는지 확인합니다. 예를 들면 다음과 같습니다.

```
(overcloud) [stack@undercloud-0 ~]$ openstack secret delete
http://10.0.0.104:9311/v1/secrets/93f62cfd-e008-401f-be74-bf057c88b04a
(overcloud) [stack@undercloud-0 ~]$ openstack secret delete
http://10.0.0.104:9311/v1/secrets/f664b5cf-5221-47e5-9887-608972a5fefb
(overcloud) [stack@undercloud-0 ~]$ openstack secret list

(overcloud) [stack@undercloud-0 ~]$
```

4.6.1.2.4. 데이터베이스 복원

controller-0 노드에 로그인한 동안 다음 단계를 실행합니다.

1. 데이터베이스 복원을 위해 ***barbican*** 사용자에게 액세스 권한을 부여하는 컨트롤러에 barbican 데이터베이스가 있는지 확인합니다.

```
[heat-admin@controller-0 ~]$ mysql -u barbican -p"seDJRsMNRrBdFryCmNUEFPPev"
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3799
Server version: 10.1.20-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
```

```
+--------------------+
| Database           |
+--------------------+
| barbican           |
| information_schema |
+--------------------+
2 rows in set (0.00 sec)

MariaDB [(none)]> exit
Bye
[heat-admin@controller-0 ~]$
```

9) 백업 파일을 **barbican** 데이터베이스에 복원합니다.

+

```
[heat-admin@controller-0 ~]$ sudo mysql -u barbican -p"seDJRsMNRrBdFryCmNUEFPPev"
barbican < barbican_db_backup.sql
[heat-admin@controller-0 ~]$
```

4.6.1.2.5. 복원 프로세스 확인

1. 오버클라우드에서 테스트 보안이 성공적으로 복원되었는지 확인합니다.

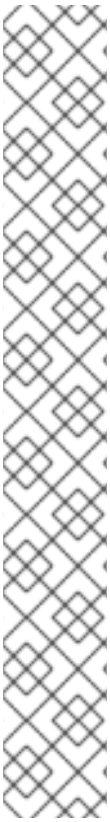
```
(overcloud) [stack@undercloud-0 ~]$ openstack secret list
+------------------------------------------------------------------------+-----------+----------------------
---+--------+----------------------------------------+-----------+------------+-------------+------+------
------+
| Secret href                                                            | Name      | Created              | Status |
Content types                        | Algorithm | Bit length | Secret type | Mode | Expiration |
+------------------------------------------------------------------------+-----------+----------------------
---+--------+----------------------------------------+-----------+------------+-------------+------+------
------+
| http://10.0.0.104:9311/v1/secrets/93f62cfd-e008-401f-be74-bf057c88b04a | testSecret |
2018-06-19T18:25:25+00:00 | ACTIVE | {u'default': u'text/plain'}              | aes       |        256 |
opaque      | cbc  | None       |
| http://10.0.0.104:9311/v1/secrets/f664b5cf-5221-47e5-9887-608972a5fefb | swift_key  |
2018-06-19T18:24:40+00:00 | ACTIVE | {u'default': u'application/octet-stream'} | aes       |
256 | symmetric   | ctr  | None       |
+------------------------------------------------------------------------+-----------+----------------------
---+--------+----------------------------------------+-----------+------------+-------------+------+------
------+
(overcloud) [stack@undercloud-0 ~]$
```

5장. CINDER 볼륨 암호화

barbican을 사용하여 Block Storage(cinder) 암호화 키를 관리할 수 있습니다. 이 구성에서는 LUKS를 사용하여 부팅 디스크를 포함하여 인스턴스에 연결된 디스크를 암호화합니다. 키 관리는 사용자에게 투명합니다. **luks** 를 암호화 유형으로 사용하여 새 볼륨을 생성하면 cinder는 볼륨에 대한 대칭 키 시크릿을 생성하고 barbican에 저장합니다. 인스턴스를 부팅하거나 암호화된 볼륨을 연결할 때 nova는 barbican에서 키를 검색하고 해당 시크릿을 Compute 노드에 Libvirt 시크릿으로 로컬로 저장합니다.

중요

암호화되지 않은 경우 처음 사용하는 동안 Nova 형식이 암호화된 볼륨입니다. 그런 다음 결과 블록 장치가 컴퓨팅 노드에 제공됩니다.



참고

구성 파일을 업데이트하려면 특정 OpenStack 서비스가 컨테이너 내에서 실행됩니다. 이 서비스는 keystone, nova, cinder에 적용됩니다. 따라서 다음을 고려해야 할 관리 사례가 있습니다.

- 물리적 노드의 호스트 운영 체제(예: **/etc/cinder/cinder.conf**)에 있는 구성 파일을 업데이트하지 마십시오. 컨테이너화된 서비스는 이 파일을 참조하지 않습니다.
- 컨테이너 내에서 실행 중인 구성 파일을 업데이트하지 마십시오. 컨테이너를 다시 시작하면 변경 사항이 손실됩니다.
  대신 컨테이너화된 서비스를 변경해야 하는 경우 컨테이너를 생성하는 데 사용되는 **/var/lib/config-data/puppet-generated/** 에서 구성 파일을 업데이트합니다.

  예를 들면 다음과 같습니다.

- keystone: **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf**
- cinder: **/var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf**
- nova: **/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf**
  변경 사항은 컨테이너를 재시작한 후 적용됩니다.

1. **cinder-volume** 및 **nova-compute** 서비스를 실행하는 노드에서 nova 및 cinder가 모두 키 관리에 barbican을 사용하도록 구성되어 있는지 확인합니다.

```
$ crudini --get /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf
key_manager backend
castellan.key_manager.barbican_key_manager.BarbicanKeyManager

$ crudini --get /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf
key_manager backend
castellan.key_manager.barbican_key_manager.BarbicanKeyManager
```

2. 암호화를 사용하는 볼륨 템플릿을 생성합니다. 새 볼륨을 만들 때 여기에 정의된 설정을 모델링할 수 있습니다.

```
$ openstack volume type create --encryption-provider
nova.volume.encryptors.luks.LuksEncryptor --encryption-cipher aes-xts-plain64 --encryption-
key-size 256 --encryption-control-location front-end LuksEncryptor-Template-256
```

```
+-------------+-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| Field       | Value
|
+-------------+-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| description | None
|
| encryption  | cipher='aes-xts-plain64', control_location='front-end', encryption_id='9df604d0-
8584-4ce8-b450-e13e6316c4d3', key_size='256',
provider='nova.volume.encryptors.luks.LuksEncryptor' |
| id          | 78898a82-8f4c-44b2-a460-40a5da9e4d59
|
| is_public   | True
|
| name        | LuksEncryptor-Template-256
|
+-------------+-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
```

3. 새 볼륨을 생성하고 **LuksEncryptor-Template-256** 설정을 사용하도록 지정합니다.

참고

암호화된 볼륨을 생성하는 사용자에게 프로젝트에 대한 **작성자** 바bican 역할이 있는지 확인합니다. 자세한 내용은 **creator 역할에 대한 사용자 액세스 권한 부여** 섹션을 참조하십시오.

```
$ openstack volume create --size 1 --type LuksEncryptor-Template-256 'Encrypted-Test-
Volume'
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2018-01-22T00:19:06.000000           |
| description         | None                                 |
| encrypted           | True                                 |
| id                  | a361fd0b-882a-46cc-a669-c633630b5c93 |
| migration_status    | None                                 |
| multiattach         | False                                |
| name                | Encrypted-Test-Volume                |
| properties          |                                      |
| replication_status  | None                                 |
| size                | 1                                    |
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | LuksEncryptor-Template-256           |
| updated_at          | None                                 |
| user_id             | 0e73cb3111614365a144e7f8f1a972af     |
+---------------------+--------------------------------------+
```

결과 시크릿은 barbican 백엔드에 자동으로 업로드됩니다.

4. barbican을 사용하여 디스크 암호화 키가 있는지 확인합니다. 이 예제에서 타임스탬프는 LUKS 볼륨 생성 시간과 일치합니다.

```
$ openstack secret list
+------------------------------------------------------------------------------+------+----------
---------+--------+-----------------------------------------+----------+-----------+-------------+------
+------------+
| Secret href                                                                  | Name | Created                | Status
| Content types                        | Algorithm | Bit length | Secret type | Mode | Expiration |
+------------------------------------------------------------------------------+------+----------
---------+--------+-----------------------------------------+----------+-----------+-------------+------
+------------+
| https://192.168.123.169:9311/v1/secrets/24845e6d-64a5-4071-ba99-0fdd1046172e | None |
2018-01-22T02:23:15+00:00 | ACTIVE | {u'default': u'application/octet-stream'} | aes       |
256 | symmetric   | None | None       |
+------------------------------------------------------------------------------+------+----------
---------+--------+-----------------------------------------+----------+-----------+-------------+------
+------------+
```

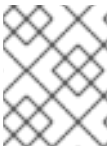
5. 기존 인스턴스에 새 볼륨을 연결합니다. 예를 들면 다음과 같습니다.

```
$ openstack server add volume testInstance Encrypted-Test-Volume
```

그런 다음 볼륨이 게스트 운영 체제에 표시되고 내장 툴을 사용하여 마운트할 수 있습니다.

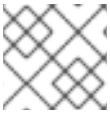
5.1. 기존 볼륨 키를 BARBICAN으로 마이그레이션

이전에는 배포에서 **ConfKeyManager** 를 사용하여 디스크 암호화 키를 관리할 수 있었습니다. 이는 고정된 키가 생성되었으며 nova 및 cinder 구성 파일에 저장되었습니다. 다음 절차를 사용하여 키 ID를 barbican으로 마이그레이션할 수 있습니다. 이 유틸리티는 barbican으로 마이그레이션을 위해 범위 내에서 **encryption_key_id** 항목의 데이터베이스를 스캔하여 작동합니다. 각 항목은 새 barbican 키 ID를 가져오고 기존 **ConfKeyManager** 시크릿은 유지됩니다.



참고

이전에는 **ConfKeyManager** 를 사용하여 암호화된 볼륨의 소유권을 다시 할당할 수 있었습니다. barbican으로 관리하는 볼륨은 사용할 수 없습니다.



참고

barbican을 활성화하면 기존 **keymgr** 볼륨이 손상되지 않습니다.

마이그레이션 프로세스가 활성화되면 마이그레이션 프로세스가 자동으로 실행되지만 다음 섹션에 설명된 일부 구성이 필요합니다. 실제 마이그레이션은 **cinder-volume** 및 **cinder-backup** 프로세스에서 실행되며 cinder 로그 파일의 진행 상황을 추적할 수 있습니다.

- **Cinder-volume - cinder** 의 볼륨 및 스냅샷 테이블에 저장된 키를 마이그레이션합니다.
- **cinder-backup** - Backups 테이블에서 키를 마이그레이션합니다.

5.1.1. 마이그레이션 단계 개요

1. barbican 서비스를 배포합니다.
2. cinder 서비스에 **creator** 역할을 추가합니다. 예를 들면 다음과 같습니다.

```
#openstack role create creator
#openstack role add --user cinder creator  --project service
```

3. **cinder-volume** 및 **cinder-backup** 서비스를 다시 시작합니다.
4. **cinder-volume** 및 **cinder-backup** 은 마이그레이션 프로세스를 자동으로 시작합니다.
5. 마이그레이션이 완료되었음을 나타내는 메시지의 로그를 모니터링하고 **ConfKeyManager** all-zeros 암호화 키 ID를 더 이상 사용하지 않는지 확인합니다.
6. **cinder.conf** 및 **nova.conf** 에서 **fixed_key** 옵션을 제거합니다. 이 설정이 구성되어 있는 노드를 확인해야 합니다.
7. cinder 서비스에서 **creator** 역할을 제거합니다.

5.1.2. 동작 차이점

Barbican-managed 암호화된 볼륨은 **ConfKeyManager** 를 사용하는 볼륨과 다르게 동작합니다.

- 현재 barbican 시크릿의 소유권을 양도할 수 없기 때문에 암호화된 볼륨에 대한 소유권을 양도할 수 없습니다.
- Barbican은 일부 cinder 볼륨 작업에 영향을 줄 수 있는 시크릿을 읽고 삭제할 수 있는 사람을 더욱 제한적입니다. 예를 들어 사용자는 다른 사용자의 볼륨을 연결, 분리 또는 삭제할 수 없습니다.

5.1.3. 마이그레이션 프로세스 검토

이 섹션에서는 마이그레이션 작업의 상태를 확인하는 방법에 대해 설명합니다. 프로세스를 시작하면 이러한 항목 중 하나가 로그에 표시됩니다. 이는 마이그레이션이 올바르게 시작되었는지 또는 발생한 문제를 식별하는지 여부를 나타냅니다.

- **ConfKeyManager가 여전히 사용 중이므로 암호화 키를 마이그레이션하지 않습니다.**
- **ConfKeyManager의 fixed_key가 사용되지 않기 때문에 암호화 키를 마이그레이션하지 않습니다.**
- **'XXX' key_manager 백엔드로 마이그레이션이 지원되지 않으므로 암호화 키를 마이그레이션하지 않습니다.** - 이 메시지는 나타나지 않습니다. 이 메시지는 barbican 이외의 다른 Key Manager 백엔드가 발생하는 코드를 처리하는 안전한 검사입니다. 이는 코드가 하나의 마이그레이션 시나리오만 지원하기 때문입니다. ConfKeyManager *에서* b *arbican에* 있습니다.
- **이 호스트와 연결된 볼륨이 없기 때문에 암호화 키를 마이그레이션하지 않습니다.** - **cinder-volume** 이 여러 호스트에서 실행되고 있을 때 발생할 수 있으며 특정 호스트에 해당 볼륨이 연결되어 있지 않습니다. 이는 모든 호스트가 자체 볼륨을 처리하므로 이러한 문제가 발생합니다.
- **ConfKeyManager 키의 마이그레이션을 시작합니다.**
- **볼륨 <UUID> 암호화 키를 Barbican로** 마이그레이션 - 마이그레이션 중에 호스트의 모든 볼륨이 검사되고 볼륨이 여전히 ConfKeyManager의 키 ID(**00000000-0000-0000-000000000000**)임을 확인하면 이 메시지가 표시됩니다.
    - **cinder-backup** 의 경우 이 메시지는 약간 다른 대문자를 사용합니다. **볼륨 마이그레이션 [...]**

- 각 호스트에서 모든 볼륨을 검사하면 호스트에 요약 상태 메시지가 표시됩니다.

  > `No volumes are using the ConfKeyManager's encryption_key_id.`
  > `No backups are known to be using the ConfKeyManager's encryption_key_id.`

  다음 항목도 볼 수 있습니다.

  ConfKeyManager의 모든 0 암호화 키 ID를 사용하는%d 볼륨이 여전히 있습니다.ConfKeyManager의 모든 0s 암호화 키 ID를 사용하는 %d 백업이 여전히 있습니다. 이러한 두 메시지 모두 **cinder-volume** 및 **cinder-backup** 로그에 표시될 수 있습니다. 각 서비스는 자체 항목의 마이그레이션만 처리하는 반면, 서비스는 다른 항목의 상태를 확인합니다. 그 결과 **cinder-volume** 은 **cinder-backup** 이 여전히 마이그레이션할 백업이 있는지 알고 있으며 **cinder-volume** 서비스에 마이그레이션할 볼륨이 있는지 여부를 알고 있습니다.

  각 호스트는 자체 볼륨만 마이그레이션하지만 요약 메시지는 여전히 마이그레이션이 필요한지 여부에 대한 글로벌 평가를 기반으로 하므로 모든 볼륨에 대한 마이그레이션이 완료되었는지 확인할 수 있습니다. 확인이 수신되면 **cinder.conf** 및 **nova.conf** 에서 **fixed_key** 설정을 제거하십시오. 자세한 내용은 *아래 고정된 키 정리* 섹션을 참조하십시오.

5.1.4. 마이그레이션 프로세스 문제 해결

5.1.4.1. 역할 할당

barbican 시크릿은 요청자에게 create 역할이 있는 경우에만 생성할 수 있습니다. **즉, cinder 서비스 자체에 생성자 역할이 필요합니다. 그렇지 않으면 다음과 유사한 로그 시퀀스가 발생합니다.**

1. **ConfKeyManager 키의 마이그레이션을 시작합니다.**
2. **볼륨 <UUID> 암호화 키를 Barbican으로 마이그레이션**
3. **error migrating encryption key: Forbidden: Secret creation attempt not allowed - 사용자/프로젝트 권한을 검토하십시오.**
4. **ConfKeyManager의 all-zeros 암호화 키 ID를 사용하여 여전히 %d 볼륨이 있습니다.**

키 메시지는 세 번째: 시크릿 생성 시도가 허용되지 않습니다. 이 문제를 해결하려면 cinder 계정의 권한을 업데이트합니다.

1. **openstack role add --project service --user cinder creator를 실행합니다.**
2. **cinder-volume 및 cinder-backup 서비스를 다시 시작합니다.**

따라서 마이그레이션 시 다음 시도가 성공해야 합니다.

5.1.5. 수정된 키를 정리합니다.

중요

encryption_key_id 는 **Queens** 릴리스의 일부로 최근 백업 테이블에 추가되었습니다. 결과적으로 암호화된 볼륨의 기존 백업이 존재할 가능성이 높습니다. **all-zeros encryption_key_id** 는 백업 자체에 저장되지만 **Backup** 데이터베이스에는 표시되지 않습니다. 따라서 암호화된 볼륨의 백업이 모두 **0**인 **ConfKeyMgr** 키 **ID**에 의존하는 암호화된 볼륨의 백업이 존재하는지에 대해 마이그레이션 프로세스에서는 알 수 없습니다.

키 **ID**를 **barbican**로 마이그레이션한 후 고정 키는 구성 파일에 유지됩니다. **fixed_key** 값이 **.conf** 파일에서 암호화되지 않으므로 일부 사용자에게 보안 문제가 표시될 수 있습니다. 이 문제를 해결하려면 **nova** 및 **cinder** 구성에서 **fixed_key** 값을 수동으로 제거할 수 있습니다. 그러나 이 값에 종속된 디스크는 여전히 액세스할 수 없기 때문에 먼저 완료하여 로그 파일의 출력을 검토합니다.

1. 기존 **fixed_key** 값을 검토합니다. 두 서비스 모두의 값과 일치해야 합니다.

```
crudini --get /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf keymgr
fixed_key
crudini --get /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf keymgr
fixed_key
```

2. 중요: 기존 **fixed_key** 값을 백업하십시오. 이렇게 하면 문제가 발생 하는 경우 값을 복원 하거나 이전 암호화 키를 사용 하는 백업을 복원 해야 합니다.**This allows you to restore the value if something goes wrong, or if you need to restore a backup that uses the old encryption key.**

3. **fixed_key** 값을 삭제합니다.

```
crudini --del /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf keymgr
fixed_key
crudini --del /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf keymgr
fixed_key
```

6장. AT-REST SWIFT 오브젝트 암호화

기본적으로 **Object Storage**에 업로드된 오브젝트는 암호화되지 않은 상태로 저장됩니다. 이로 인해 파일 시스템에서 개체에 직접 액세스할 수 있습니다. 이는 디스크가 삭제되기 전에 제대로 지워지지 않으면 보안 위험이 발생할 수 있습니다. **barbican**을 활성화하면 **Object Storage** 서비스**(swift)**에서 저장된 오브젝트를 투명하게 암호화하고 해독할 수 있습니다. 유휴 암호화는 디스크에 저장되는 동안 암호화되는 개체를 나타내는 전송 내 암호화와 다릅니다.

Swift는 **swift**로 업로드할 때 자동으로 암호화되고 사용자에게 제공할 때 자동으로 암호 해독되는 오브젝트와 함께 이러한 암호화 작업을 투명하게 수행합니다. 이 암호화 및 암호 해독은 **barbican**에 저장된 동일한 **(symmetric)** 키를 사용하여 수행됩니다.

참고

이제 데이터가 암호화된 상태에 저장되므로 암호화를 비활성화하고 **swift** 클러스터에 데이터를 추가한 후에는 암호화를 비활성화할 수 없습니다. 따라서 동일한 키로 암호화를 다시 활성화할 때까지 암호화가 비활성화된 경우 데이터를 읽을 수 없습니다.

6.1. SWIFT에 대해 유휴 상태의 암호화 활성화

1. 사용자 환경 파일에 **SwiftEncryptionEnabled: True** 를 포함하여 **swift** 암호화 기능을 활성화한 다음 **/home/stack/overcloud_deploy.sh** 를 사용하여 **openstack overcloud deploy** 를 다시 실행할 수 있습니다. ***Barbican** 설치 장에 설명된 대로 **barbican**을 계속 활성화해야 합니다*.

2. 복구 시 암호화를 사용하도록 **swift**가 구성되었는지 확인합니다.

```
$ crudini --get /var/lib/config-data/puppet-generated/swift/etc/swift/proxy-server.conf pipeline-main pipeline

pipeline = catch_errors healthcheck proxy-logging cache ratelimit bulk tempurl formpost authtoken keystone staticweb copy container_quotas account_quotas slo dlo versioned_writes kms_keymaster encryption proxy-logging proxy-server
```

결과에는 암호화에 대한 항목이 포함되어야 합니다.

7장. GLANCE 이미지 검증

Barbican을 활성화하면 업로드된 이미지가 변경되지 않았는지 확인하도록 **Image** 서비스(**glance**)를 구성할 수 있습니다. 이 구현에서 이미지는 먼저 **barbican**에 저장된 키로 서명됩니다. 그런 다음 이미지는 서명 정보와 함께 **glance**에 업로드됩니다. 결과적으로 각 사용 전에 이미지의 서명이 확인되고 서명이 일치하지 않으면 인스턴스 빌드 프로세스가 실패합니다.

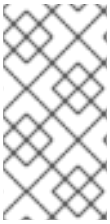
Barbican의 **glance**와의 통합은 **openssl** 명령을 개인 키와 함께 사용하여 이미지를 업로드하기 전에 **glance** 이미지에 서명할 수 있음을 의미합니다.

7.1. GLANCE 이미지 검증 활성화

환경 파일에서 **VerifyGlanceSignatures: True** 설정을 사용하여 이미지 확인을 활성화합니다. 이 설정을 적용하려면 **openstack overcloud deploy** 명령을 다시 실행해야 합니다.

Glance 이미지 검증이 활성화되었는지 확인하려면 오버클라우드 컴퓨팅 노드에서 다음 명령을 실행합니다.

```
$ sudo crudini --get /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf glance
verify_glance_signatures
```



참고

Ceph를 이미지 및 **Compute** 서비스의 백엔드로 사용하면 **CoW** 복제본이 생성됩니다. 따라서 이미지 서명 확인을 수행할 수 없습니다.

7.2. 이미지 검증

검증을 위해 **Glance** 이미지를 구성하려면 다음 단계를 완료합니다.

1. **barbican**을 사용하도록 **glance**가 구성되었는지 확인합니다.

```
$ sudo crudini --get /var/lib/config-data/puppet-generated/glance_api/etc/glance/glance-
api.conf key_manager backend
castellan.key_manager.barbican_key_manager.BarbicanKeyManager
```

2. 인증서를 생성합니다.

```
openssl genrsa -out private_key.pem 1024
openssl rsa -pubout -in private_key.pem -out public_key.pem
openssl req -new -key private_key.pem -out cert_request.csr
openssl x509 -req -days 14 -in cert_request.csr -signkey private_key.pem -out
x509_signing_cert.crt
```

3. **barbican** 시크릿 저장소에 인증서를 추가합니다.

```
$ source ~/overcloudrc
$ openstack secret store --name signing-cert --algorithm RSA --secret-type certificate --
payload-content-type "application/octet-stream" --payload-content-encoding base64  --
payload "$(base64 x509_signing_cert.crt)" -c 'Secret href' -f value
https://192.168.123.170:9311/v1/secrets/5df14c2b-f221-4a02-948e-48a61edd3f5b
```

참고

이후 단계에서 사용할 수 있도록 결과 **UUID**를 기록합니다. 이 예에서 인증서의 **UUID**는 **5df14c2b-f221-4a02-948e-48a61edd3f5b** 입니다.

4. **private_key.pem** 을 사용하여 이미지에 서명하고 **.signature** 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
$ openssl dgst -sha256 -sign private_key.pem -sigopt rsa_padding_mode:pss -out cirros-
0.4.0.signature cirros-0.4.0-x86_64-disk.img
```

5. 결과 **.signature** 파일을 ***base64*** 형식으로 변환합니다.

```
$ base64 -w 0 cirros-0.4.0.signature  > cirros-0.4.0.signature.b64
```

6. ***base64*** 값을 후속 명령에서 사용하도록 변수에 로드합니다.

```
$ cirros_signature_b64=$(cat cirros-0.4.0.signature.b64)
```

7. 서명된 이미지를 **glance**에 업로드합니다. **img_signature_certificate_uuid** 의 경우 이전에 **barbican**에 업로드한 서명 키의 **UUID**를 지정해야 합니다.

```
openstack image create \
--container-format bare --disk-format qcow2 \
--property img_signature="$cirros_signature_b64" \
--property img_signature_certificate_uuid="5df14c2b-f221-4a02-948e-48a61edd3f5b" \
--property img_signature_hash_method="SHA-256" \
```

```
 --property img_signature_key_type="RSA-PSS" cirros_0_4_0_signed \
 --file cirros-0.4.0-x86_64-disk.img
+---------------------------------+------------------------------------------------------------------------------
----+
| Property                        | Value                                                                        |
+---------------------------------+------------------------------------------------------------------------------
----+
| checksum                        | 25c96942084f61e008d593b6c2cfda00
|
| container_format                | bare                                                                         |
| created_at                      | 2018-01-23T05:37:31Z                                                         |
| disk_format                     | qcow2                                                                        |
| id                              | d3396fa0-2ea2-4832-8a77-d36fa3f2ab27                                         |
| img_signature                   |
lcI7nGgoKxnCyOcsJ4abbEZEpzXByFPIgiPeiT+Otjz0yvW00KNN3fI0AA6tn9EXrp7fb2xBDE4UaO3v |
|                                 |
IFquV/s3mU4LcCiGdBAl3pGsMlmZZIQFVNcUPOaayS1kQYKY7kxYmU9iq/AZYyPw37KQI52smC/zoO54 |
|                                 | zZ+JpnfwIsM=                                                                 |
| img_signature_certificate_uuid | ba3641c2-6a3d-445a-8543-851a68110eab
|
| img_signature_hash_method       | SHA-256                                                                      |
| img_signature_key_type          | RSA-PSS                                                                      |
| min_disk                        | 0                                                                            |
| min_ram                         | 0                                                                            |
| name                            | cirros_0_4_0_signed                                                          |
| owner                           | 9f812310df904e6ea01e1bacb84c9f1a
|
| protected                       | False                                                                        |
| size                            | None                                                                         |
| status                          | active                                                                       |
| tags                            | []                                                                           |
| updated_at                      | 2018-01-23T05:37:31Z                                                         |
| virtual_size                    | None                                                                         |
| visibility                      | shared                                                                       |
+---------------------------------+------------------------------------------------------------------------------
----+
```

8. **Glance의 이미지 유효성 검사 활동은 Compute 로그: /var/log/containers/nova/nova-compute.log 에서 확인할 수 있습니다. 예를 들어 인스턴스가 부팅될 때 다음 항목을 예상할 수 있습니다.**

```
2018-05-24 12:48:35.256 1 INFO nova.image.glance [req-7c271904-4975-4771-9d26-
cbea6c0ade31 b464b2fd2a2140e9a88bbdacf67bdd8c a3db2f2beaee454182c95b646fa7331f
- default default] Image signature verification succeeded for image d3396fa0-2ea2-4832-
8a77-d36fa3f2ab27
```