Red Hat OpenStack Platform 17.1

키 관리자 서비스를 사용하여 보안 관리

OpenStack 배포와 키 관리자 서비스(barbican) 통합.

Last Updated: 2024-09-27

Red Hat OpenStack Platform 17.1 키 관리자 서비스를 사용하여 보안 관리

OpenStack 배포와 키 관리자 서비스(barbican) 통합.

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

OpenStack Key Manager(barbican)를 OpenStack 배포와 통합하는 방법.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체 3

RED HAT 문서에 관한 피드백 제공 4

1장. OPENSTACK KEY MANAGER 배포 및 구성(BARBICAN) 5
1.1. OPENSTACK KEY MANAGER 워크플로 5
1.2. OPENSTACK KEY MANAGER 암호화 유형 6
1.3. 키 관리자 배포 7
1.4. 키 관리자 정책 보기 10

2장. OPENSTACK KEY MANAGER(BARBICAN)를 사용하여 시크릿 및 키 관리 13
2.1. 보안 보기 13
2.2. 시크릿 생성 13
2.3. 보안에 페이로드 추가 14
2.4. 보안 삭제 14
2.5. 대칭 키 생성 14
2.6. 간단한 암호화 암호화 키 백업 16
2.7. 백업에서 간단한 암호화 암호화 키 복원 19

3장. HSM(HARDWARE SECURITY MODULE) 어플라이언스와 OPENSTACK KEY MANAGER(BARBICAN) 통합 21
3.1. OPENSTACK KEY MANAGER(BARBICAN)와 ATOS HSM 통합 21
3.2. OPENSTACK KEY MANAGER(BARBICAN)와 THALES LUNA NETWORK HSM 통합 24
3.3. OPENSTACK KEY MANAGER(BARBICAN)와 ENTRUST NSHIELD CONNECT XC HSM 통합 26
3.4. MKEK 및 HMAC 키 교체 30

4장. OPENSTACK 서비스 암호화 및 검증 32
4.1. OBJECT STORAGE(SWIFT) AT-REST 오브젝트 암호화 32
4.2. BLOCK STORAGE(CINDER) 볼륨 암호화 33
4.3. BLOCK STORAGE(CINDER) 볼륨 이미지 검증 41
4.4. IMAGE 서비스(GLANCE) 이미지에 서명 44
4.5. 스냅샷 검증 47

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.

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 어떻게 더 나은지 알려주십시오.

Jira에서 문서 피드백 제공

문제 생성 양식을 사용하여 문서에 대한 피드백을 제공합니다. Jira 문제는 Red Hat OpenStack Platform Jira 프로젝트에서 생성되어 피드백의 진행 상황을 추적할 수 있습니다.

1. Jira에 로그인했는지 확인합니다. Jira 계정이 없는 경우 피드백을 제출할 계정을 생성합니다.
2. 다음 링크를 클릭하여 **문제 생성** 페이지를 엽니다.
   https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12336920&summary=Documentation%20feedback:%20%3CAdd%20summary%20here%3E&i
   <Include+the+documentation+URL,+the%20chapter+or+section+number,+and+a+detailed+descrip
3. **요약** 및 **설명** 필드를 작성합니다. **설명** 필드에 문서 URL, 장 또는 섹션 번호, 문제에 대한 자세한 설명을 포함합니다. 양식의 다른 필드를 수정하지 마십시오.
4. **생성**을 클릭합니다.

1장. OPENSTACK KEY MANAGER 배포 및 구성(BARBICAN)

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. Red Hat OpenStack Platform에서는 Barbican이 기본적으로 활성화되어 있지 않습니다. 기존 OpenStack 배포에 barbican을 배포할 수 있습니다.

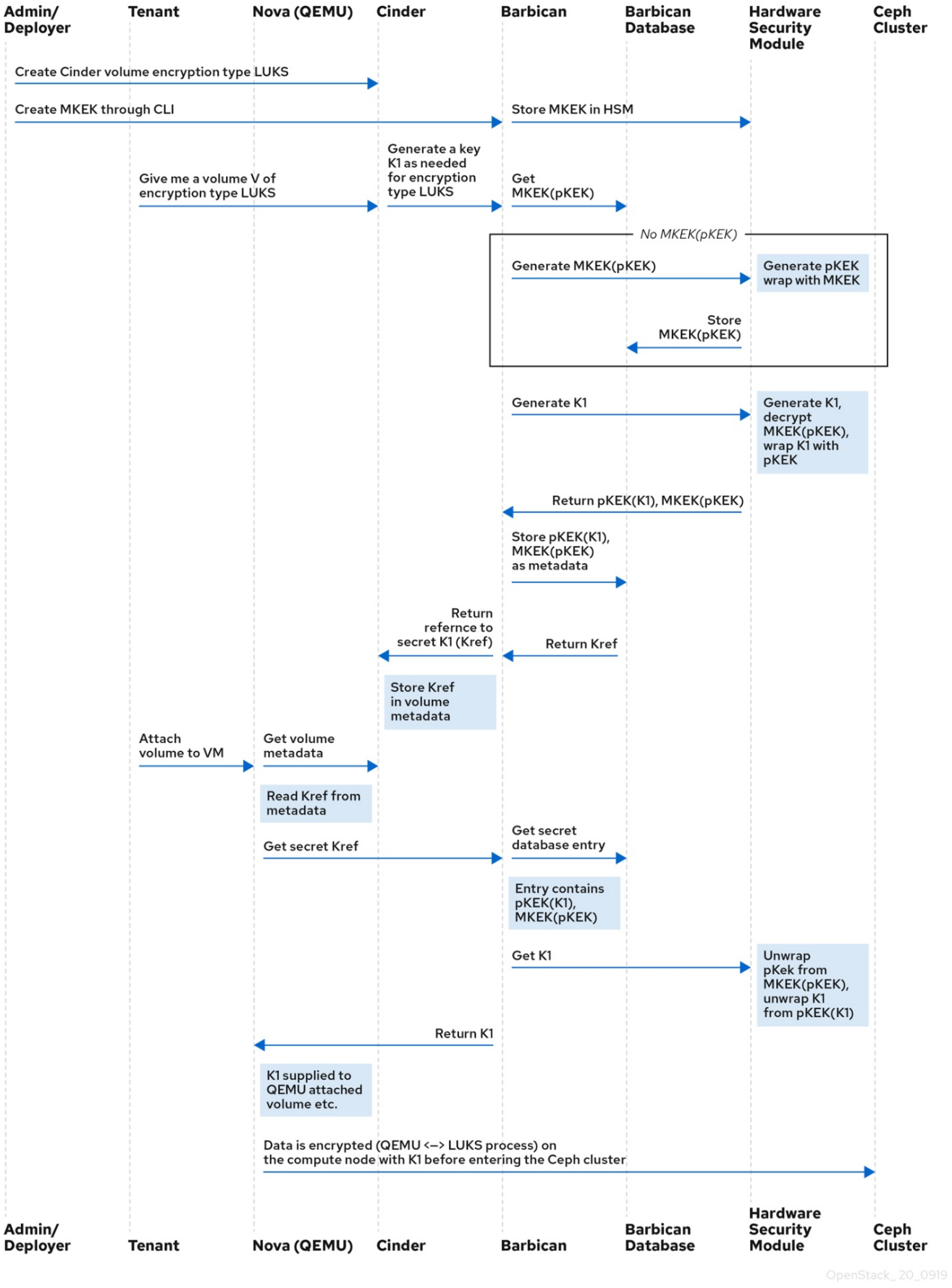
Barbican은 현재 이 가이드에 설명된 다음 사용 사례를 지원합니다.

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

OpenStack Key Manager는 Block Storage(cinder), 네트워킹(neutron) 및 Compute(nova) 구성 요소와 통합됩니다.

1.1. OPENSTACK KEY MANAGER 워크플로

다음 다이어그램에서는 OpenStack Key Manager에서 사용자 환경의 시크릿을 관리하는 데 사용하는 워크플로를 보여줍니다.



1.2. OPENSTACK KEY MANAGER 암호화 유형

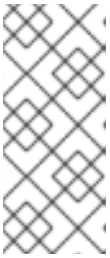
인증서, API 키 및 암호와 같은 시크릿은 barbican 데이터베이스의 암호화된 Blob에 저장하거나 보안 스토리지 시스템에 직접 저장할 수 있습니다. 간단한 암호화 플러그인 또는 PKCS#11 암호화 플러그인을 사용하여 시크릿을 암호화할 수 있습니다.

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

- **간단한 암호화 플러그인** - 간단한 암호화 플러그인은 기본적으로 활성화되어 있으며 단일 대칭 키를 사용하여 모든 시크릿 페이로드를 암호화합니다. 이 키는 **barbican.conf** 파일의 일반 텍스트에 저장되므로 이 파일에 대한 무단 액세스를 방지하는 것이 중요합니다.

- **PKCS#11 암호화 플러그인** - PKCS#11 암호화 플러그인은 barbican 데이터베이스에 저장된 프로젝트별 키 암호화 키(pKEK)로 시크릿을 암호화합니다. 이러한 프로젝트별 pKEK는 HSM(하드웨어 보안 모듈)에 저장된 MKEK(키 암호화 키)로 암호화됩니다. 모든 암호화 및 암호 해독 작업은 프로세스 내 메모리가 아닌 HSM에서 수행됩니다. PKCS#11 플러그인은 PKCS#11 API를 통해 HSM과 통신합니다. 암호화는 보안 하드웨어에서 수행되며 프로젝트당 다른 pKEK가 사용되므로 이 옵션은 간단한 암호화 플러그인보다 안전합니다.
  Red Hat은 다음 HSM과 함께 PKCS#11 백엔드를 지원합니다.

장치	릴리스에서 지원됨	HA(고가용성) 지원
ATOS Trustway Proteccio NetHSM	16.0+	16.1+
신뢰할 수 있는 nShield Connect HSM	16.0+	지원되지 않음
Thales Luna Network HSM	16.1+(기술 프리뷰)	16.1+(기술 프리뷰)



참고

HA(고가용성) 옵션 관련: barbican 서비스는 Apache 내에서 실행되며 고가용성을 위해 HAProxy를 사용하도록 director에 의해 구성됩니다. 백엔드 계층의 HA 옵션은 사용 중인 백엔드에 따라 달라집니다. 예를 들어 간단한 암호화의 경우 모든 barbican 인스턴스에 config 파일에 동일한 암호화 키가 있어 간단한 HA 구성이 생성됩니다.

1.2.1. 여러 암호화 메커니즘 구성

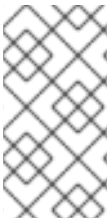
두 개 이상의 백엔드를 사용하도록 Barbican의 단일 인스턴스를 구성할 수 있습니다. 이 작업이 완료되면 백엔드를 **글로벌 기본** 백엔드로 지정해야 합니다. 프로젝트당 기본 백엔드를 지정할 수도 있습니다. 프로젝트에 대한 매핑이 없는 경우 해당 프로젝트의 시크릿은 글로벌 기본 백엔드를 사용하여 저장됩니다.

예를 들어 Simple crypto 및 PKCS#11 플러그인을 모두 사용하도록 Barbican을 구성할 수 있습니다. Simple crypto를 글로벌 기본값으로 설정하면 모든 프로젝트에서 해당 백엔드를 사용합니다. 그런 다음 PKCS#11을 해당 프로젝트의 기본 백엔드로 설정하여 PKCS#11 백엔드를 사용하는 프로젝트를 지정할 수 있습니다.

새 백엔드로 마이그레이션하려면 새 백엔드를 글로벌 기본값으로 활성화하거나 프로젝트별 백엔드로 활성화하는 동안 원본을 계속 사용할 수 있습니다. 결과적으로 이전 백엔드를 통해 이전 시크릿을 사용할 수 있으며 새 시크릿은 새로운 글로벌 기본 백엔드에 저장됩니다.

1.3. 키 관리자 배포

OpenStack Key Manager를 배포하려면 먼저 barbican 서비스에 대한 환경 파일을 생성하고 추가 환경 파일을 사용하여 오버클라우드를 재배포합니다. 그런 다음 **생성자** 역할에 사용자를 추가하여 barbican 보안을 생성 및 편집하거나 시크릿을 barbican에 저장하는 암호화된 볼륨을 생성합니다.



참고

이 절차에서는 **simple_crypto** 백엔드를 사용하도록 barbican을 구성합니다. 다른 구성이 필요한 **PKCS#11** 및 HSM 사용 방법에 따라 다른 heat 템플릿 파일과 같은 추가 백엔드를 사용할 수 있습니다. KMIP, Hashicorp Vault 및 DogTag와 같은 기타 백엔드는 지원되지 않습니다.

사전 요구 사항

- 오버클라우드가 배포 및 실행 중

절차

1. 언더클라우드 노드에서 barbican의 환경 파일을 만듭니다.

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

BarbicanSimpleCryptoGlobalDefault 는 이 플러그인을 글로벌 기본 플러그인으로 설정합니다.

환경 파일에 다음 옵션을 추가할 수도 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. 해시 값을 사용합니다(예: **{ barbican-context_is_admin: { key: context_is_admin, value: 'role:admin' } }** }. 그러면 이 항목이 **/etc/barbican/policy.json** 에 추가됩니다. 정책은 이후 섹션에서 자세히 설명합니다.
- **BarbicanSimpleCryptoKek** - none이 지정되지 않은 경우 director에서 KEK(Key Encryption Key)를 생성합니다.

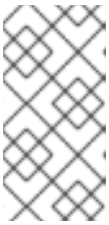
2. 스크립트에서 이전에 추가한 역할, 템플릿 또는 환경 파일을 제거하지 않고 **openstack overcloud deploy** 명령에 다음 파일을 추가합니다.
   - /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml
   - /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-simple-crypto.yaml
   - /home/stack/templates/configure-barbican.yaml
3. 배포 스크립트를 다시 실행하여 배포에 변경 사항을 적용합니다.

```
$ 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/containers-prepare-parameter.yaml \
    -e /home/stack/templates/config_lvm.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
    -e /home/stack/templates/network/network-environment.yaml \
    -e /home/stack/templates/hostnames.yml \
    -e /home/stack/templates/nodes_data.yaml \
    -e /home/stack/templates/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/templates/configure-barbican.yaml \
    --log-file overcloud_deployment_38.log
```

4. **작성자** 역할의 ID를 검색합니다.

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



참고

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

5. **사용자를 작성자 역할에 할당하고 관련 프로젝트를 지정합니다. 이 예에서는 project_a 프로젝트에서 user1 이라는 사용자가 작성자 역할에 추가됩니다.**

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

검증

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 시크릿의 페이로드를 검색합니다.

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

1.4. 키 관리자 정책 보기

Barbican은 정책을 사용하여 키 추가 또는 삭제와 같은 시크릿에 대해 작업을 수행할 수 있는 사용자를 결정합니다. 이러한 제어를 구현하기 위해 이전에 만든 생성자 와 같은 **keystone** 프로젝트 역할은 **barbican** 내부 권한에 매핑됩니다. 결과적으로 해당 프로젝트 역할에 할당된 사용자에게는 해당 **barbican** 권한이 부여됩니다.

기본 정책은 코드로 정의되며 일반적으로 수정 사항이 필요하지 않습니다. 정책 변경이 이루어지지 않은 경우 해당 환경의 기존 컨테이너를 사용하여 기본 정책을 볼 수 있습니다. 기본 정책을 변경하고 기본 값을 표시하려면 별도의 시스템을 사용하여 **openstack-barbican-api** 컨테이너를 먼저 가져옵니다.

사전 요구 사항

- **OpenStack Key Manager**가 배포 및 실행 중

절차

1. **Red Hat** 인증 정보를 사용하여 **podman**에 로그인합니다.

```
podman login
username: ********
password: ********
```

2. **openstack-barbican-api** 컨테이너를 가져옵니다.

```
podman pull \
registry.redhat.io/rhosp-rhel8/openstack-barbican-api:17.1
```

3. 현재 작업 디렉터리에 정책 파일을 생성합니다.

```
podman run -it \
registry.redhat.io/rhosp-rhel8/openstack-barbican-api:17.1 \
oslopolicy-policy-generator \
--namespace barbican > barbican-policy.yaml
```

검증

barbican-policy.yaml 파일을 검토하여 **barbican**에서 사용하는 정책을 확인합니다. 이 정책은 사용자가 시크릿 및 시크릿 메타데이터와 상호 작용하는 방법을 정의하는 네 가지 역할로 구현됩니다. 사용자는 특정 역할에 할당되어 이러한 권한을 받습니다.

admin

admin 역할은 모든 프로젝트의 시크릿을 읽고, 만들고, 편집하고, 삭제할 수 있습니다.

Creator

작성자 역할은 작성자의 범위를 지정하는 프로젝트에 있는 시크릿을 읽고, 만들고, 편집하고, 삭제할 수 있습니다.

관찰자

관찰자 역할은 비밀을 읽을 수 있습니다.

audit

감사 역할은 메타데이터만 읽을 수 있습니다. 감사 역할은 시크릿을 읽을 수 없습니다.

예를 들어 다음 항목은 각 프로젝트에 대한 **admin , observed , creator keystone** 역할을 나열합니다. 오른쪽에는 **role:admin,role:observer, role:creator** 권한이 할당됩니다.

```
#
```

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

#
#"observer": "role:observer"

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

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

파일에 **secret:put** 및 **secret:delete** 작업이 있습니다. 오른쪽에는 이러한 작업을 실행할 수 있는 권한이 있는 역할을 확인합니다. 다음 예에서 **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 있습니다. 프로젝트 일치는 **secret_project_match** 규칙에 의해 정의되며, 정책에도 정의되어 있습니다.

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

2장. OPENSTACK KEY MANAGER(BARBICAN)를 사용하여 시크릿 및 키 관리

OpenStack Key Manager를 사용하여 시크릿 및 암호화 키를 생성, 업데이트 및 삭제합니다. 암호화 키와 **barbican** 데이터베이스를 백업하고 복원할 수도 있습니다. 암호화 키와 **barbican** 데이터베이스를 정기적으로 백업하는 것이 좋습니다.

2.1. 보안 보기

시크릿 목록을 보려면 **openstack secret list** 명령을 실행합니다. 목록에는 **URI**, 이름, 유형 및 시크릿에 대한 기타 정보가 포함됩니다.

절차

- 보안 목록을 확인합니다.

```
$ 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       |
+------------------------------------------------------------------------------+------+------------------
---------+--------+-----------------------------------------+----------+------------+-------------+------
+------------+
```

2.2. 시크릿 생성

보안을 생성하려면 **openstack secret store** 명령을 실행하고 시크릿의 이름과 선택적으로 시크릿 페이로드를 지정합니다.

절차

- 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                                                                        |
+---------------+-----------------------------------------------------------------------------+
```

2.3. 보안에 페이로드 추가

시크릿의 페이로드를 변경할 수 없지만 페이로드를 지정하지 않고 시크릿을 생성한 경우 **openstack secret update** 명령을 사용하여 페이로드를 추가할 수 있습니다.

절차

- 보안에 페이로드를 추가합니다.

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

2.4. 보안 삭제

보안을 삭제하려면 **openstack secret delete** 명령을 실행하고 시크릿 **URI**를 지정합니다.

절차

- 지정된 **URI**를 사용하여 보안을 삭제합니다.

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

2.5. 대칭 키 생성

대칭 키를 생성하려면 **order create** 명령을 사용하여 키를 **barbican**에 저장합니다. 그런 다음 **nova** 디스크 암호화 및 **swift** 오브젝트 암호화와 같은 특정 작업에 대칭 키를 사용할 수 있습니다.

사전 요구 사항

- **OpenStack Key Manager**가 설치 및 실행 중

절차

1. 주문 **create** 를 사용하여 새 **256**비트 키를 생성하고 **barbican**에 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                                                                          |
+----------------+-------------------------------------------------------------------------------+
```

또한 **--mode** 옵션을 사용하여 **ctr** 또는 **cbc** 와 같은 특정 모드를 사용하도록 생성된 키를 구성할 수도 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                                                                        |
```

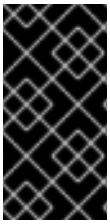
```
| Error code     | None                                                                                          |
| Error message  | None                                                                                          |
+----------------+-----------------------------------------------------------------------------------------------+
```

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                                                                                          |
+---------------+-----------------------------------------------------------------------------------------------+
```

2.6. 간단한 암호화 암호화 키 백업

간단한 암호화 암호화 키를 백업하려면 기본 **KEK**가 포함된 **barbican.conf** 파일을 보안 강화 위치로 백업한 다음 **barbican** 데이터베이스를 백업합니다.



중요

이 절차에는 테스트 시크릿 및 키를 생성하는 단계가 포함되어 있습니다. 시크릿 키를 이미 생성한 경우 테스트 키 단계를 건너뛰고 생성한 키를 사용합니다.

사전 요구 사항

- **OpenStack Key Manager**가 설치 및 실행 중
- **KEK** 백업에 대해 강화된 보안 위치가 있습니다.

절차

1. **오버클라우드에서 새 256비트 키를 생성하여 barbican에 저장합니다.**

```
(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. **테스트 보안이 생성되었는지 확인합니다.**

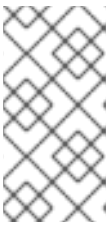
```
(overcloud) [stack@undercloud-0 ~]$ openstack secret list
+-------------------------------------------------------------------+-----------+-----------------------
---+--------+-----------------------------------------+----------+-----------+------------+------+------
------+
| Secret href                                                       | 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. **기본 KEK가 포함된 barbican.conf 파일을 보안 강화 위치로 복사합니다.**

5. **controller-0 노드에 로그인하고 *barbican* 사용자 암호를 검색합니다.**

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



참고

사용자 *barbican* 만 *barbican* 데이터베이스에 액세스할 수 있습니다. 따라서 데이터베이스를 백업하거나 복원하려면 barbican 사용자 암호가 필요합니다.

6. ***barbican* 데이터베이스를 백업합니다.**

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

7. **데이터베이스 백업이 /home/tripleo-admin:에 저장되어 있는지 확인합니다.**

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

8. **오버클라우드에서 이전에 생성한 시크릿을 삭제하고 더 이상 존재하지 않는지 확인합니다.**

```
(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 ~]$
```

2.7. 백업에서 간단한 암호화 암호화 키 복원

백업에서 **barbican** 데이터베이스를 복원하려면 **barbican** 권한이 있는 컨트롤러 노드에 로그인하고 **barbican** 데이터베이스를 복원합니다. 백업에서 **KEK**를 복원하려면 **barbican.conf** 파일을 백업 파일로 재정의합니다.

사전 요구 사항

- **OpenStack Key Manager**가 설치 및 실행 중
- **barbican.conf** 파일과 **barbican** 데이터베이스의 기존 백업이 있습니다.

절차

1. **controller-0** 노드에 로그인하고 컨트롤러에 **barbican** 사용자에게 데이터베이스를 복원할 수 있는 액세스 권한을 부여하는 **barbican** 데이터베이스가 있는지 확인합니다.

```
[tripleo-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
[tripleo-admin@controller-0 ~]$
```

2. 백업 파일을 **barbican** 데이터베이스로 복원합니다.

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

3. **이전에 백업한 파일로 barbican.conf 파일을 재정의합니다.**

검증

- **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 ~]$
```

3장. HSM(HARDWARE SECURITY MODULE) 어플라이언스와 OPENSTACK KEY MANAGER(BARBICAN) 통합

Red Hat OpenStack Platform 배포를 HSM(하드웨어 보안 모듈) 어플라이언스와 통합하여 하드웨어 기반 암호화 처리를 사용하여 보안 상태를 개선합니다. OpenStack Key Manager와 HSM 어플라이언스의 통합을 계획할 때 지원되는 암호화 유형 및 HSM 어플라이언스를 선택하고, 일반 백업을 설정하고, 배포에 영향을 줄 수 있는 기타 정보 또는 제한 사항을 검토해야 합니다.

3.1. OPENSTACK KEY MANAGER(BARBICAN)와 ATOS HSM 통합

PKCS#11 백엔드를 Trustway Proteccio Net HSM 어플라이언스와 통합하려면 매개변수가 포함된 구성 파일을 생성하여 barbican을 HSM에 연결합니다. atos_hsms 매개변수 아래에 두 개 이상의 HSM을 나열하여 HA를 활성화할 수 있습니다.

계획

기본적으로 HSM에는 최대 32개의 동시 연결이 있을 수 있습니다. 이 수를 초과하면 PKCS#11 클라이언트에서 메모리 오류가 발생할 수 있습니다. 다음과 같이 연결 수를 계산할 수 있습니다.

- 각 컨트롤러에는 하나의 barbican-api 및 하나의 barbican-worker 프로세스가 있습니다.
- 각 Barbican API 프로세스는 N Apache 작업자(기본값은 CPU 수)로 실행됩니다.
- 각 작업자는 HSM에 대한 하나의 연결이 있습니다.

각 barbican-worker 프로세스에는 데이터베이스에 대한 하나의 연결이 있습니다. BarbicanWorkers heat 매개변수를 사용하여 각 API 프로세스의 Apache 작업자 수를 정의할 수 있습니다. 기본적으로 Apache 작업자 수는 CPU 수와 일치합니다.

예를 들어 각각 32개의 코어가 있는 컨트롤러 3개가 있는 경우 각 컨트롤러의 Barbican API는 32개의 Apache 작업자를 사용합니다. 결과적으로 하나의 컨트롤러는 사용 가능한 모든 32 HSM 연결을 사용합니다. 이러한 경합을 방지하려면 각 노드에 구성된 Barbican Apache 작업자 수를 제한합니다. 이 예에서는 세 컨트롤러 모두 HSM에 대해 10 개의 동시 연결을 만들 수 있도록 BarbicanWorkers 를 10으로 설정합니다.

사전 요구 사항

- **Atos HSM에 대한 공급 업체 소프트웨어를 제공하는 암호로 보호되는 HTTPS 서버**

표 3.1. HTTPS 서버에서 제공하는 파일

파일	예제	제공자
Proteccio 클라이언트 소프트웨어 ISO 이미지 파일	Proteccio1.09.05.iso	HSM 벤더
SSL 서버 인증서	proteccio.CRT	HSM 관리자
SSL 클라이언트 인증서	client.CRT	HSM 관리자
SSL 클라이언트 키	client.KEY	HSM 관리자

절차

1. **Barbican에 대한 configure-barbican.yaml 환경 파일을 생성하고 다음 매개변수를 추가합니다.**

```
parameter_defaults
  BarbicanSimpleCryptoGlobalDefault: false
  BarbicanPkcs11CryptoGlobalDefault: true
  BarbicanPkcs11CryptoLogin: ********
  BarbicanPkcs11CryptoSlotId: 1
  ATOSVars:
    atos_client_iso_name: Proteccio1.09.05.iso
    atos_client_iso_location: https://user@PASSWORD:example.com/Proteccio1.09.05.iso
    atos_client_cert_location: https://user@PASSWORD:example.com/client.CRT
    atos_client_key_location: https://user@PASSWORD:example.com/client.KEY
    atos_hsms:
      - name: myHsm1
        server_cert_location: https://user@PASSWORD:example.com/myHsm1.CRT
        ip: 192.168.1.101
      - name: myHsm2
        server_cert_location: https://user@PASSWORD:example.com/myHsm2.CRT
        ip: ip: 192.168.1.102
```

참고

atos_hsms 매개변수는 더 이상 사용되지 않고 향후 릴리스에서 제거될 atos_hsm_ip_address 및 atos_server_cert_location 매개변수를 대체합니다.

표 3.2. heat 매개변수

매개변수	현재의
BarbicanSimpleCryptoGlobalDefault	**simplecrypto** 가 글로벌 기본값인지 여부를 결정하는 부울입니다.
BarbicanPkcs11GlobalDefault	**PKCS#11** 이 글로벌 기본값인지 여부를 결정하는 부울입니다.
BarbicanPkcs11CryptoSlotId	Barbican에서 사용할 Virtual HSM의 슬롯 ID입니다.
ATOSVars	
atos_client_iso_name	Atos 클라이언트 소프트웨어 ISO의 파일 이름입니다. 이 값은 **atos_client_iso_location** 매개변수의 URL의 파일 이름과 일치해야 합니다.
atos_client_iso_location	사용자 이름 및 암호를 포함한 URL은 Proteccio 클라이언트 소프트웨어 ISO 이미지의 HTTPS 서버 위치를 지정합니다.
atos_client_cert_location	사용자 이름 및 암호를 포함한 URL은 SSL 클라이언트 인증서의 HTTPS 서버 위치를 지정합니다.
atos_client_key_location	사용자 이름 및 암호를 포함한 URL은 SSL 클라이언트 키의 HTTPS 서버 위치를 지정합니다. 이는 위의 클라이언트 인증서와 일치하는 키여야 합니다.
atos_hsms	HSM의 이름, 인증서 위치 및 IP 주소를 지정하는 하나 이상의 HSM 목록입니다. 이 목록에 두 개 이상의 HSM을 포함하는 경우 Barbican은 부하 분산 및 고가용성을 위해 HSM을 구성합니다.

2. **배포 명령에 사용자 지정 configure-barbican.yaml , barbican.yaml 및 ATOS 특정 barbican-backend-pkcs11-atos.yaml 환경 파일을 포함하고 배포와 관련된 기타 환경 파일을 포함합니다.**

```
$ 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/containers-prepare-parameter.yaml \
  -e /home/stack/templates/config_lvm.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /home/stack/templates/network/network-environment.yaml \
  -e /home/stack/templates/hostnames.yml \
```

```
    -e /home/stack/templates/nodes_data.yaml \
    -e /home/stack/templates/extra_templates.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-pkcs11-
atos.yaml \
    -e /home/stack/templates/configure-barbican.yaml \
    --log-file overcloud_deployment_with_atos.log
```

검증

1. 테스트 보안을 생성합니다.

```
$ 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 시크릿의 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. OPENSTACK KEY MANAGER(BARBICAN)와 THALES LUNA NETWORK HSM 통합

하드웨어 기반 암호화 처리를 위해 **PKCS#11** 백엔드를 **Thales Luna Network HSM** 어플라이언스와 통합하려면 **Ansible** 역할을 사용하여 컨트롤러에 **Thales Luna** 클라이언트 소프트웨어를 다운로드 및 설치하고 사전 정의된 **HSM IP** 및 인증 정보를 포함하는 키 관리자 구성 파일을 만듭니다.

사전 요구 사항

-

Thales Luna Network HSM에 대한 벤더 소프트웨어를 제공하는 암호로 보호된 HTTPS 서버입니다.

- **이 벤더는 압축된 zip 아카이브에 Luna Network HSM 클라이언트 소프트웨어를 제공했습니다.**

절차

1. **director에 ansible-role-lunasa-hsm 역할을 설치합니다.**

```
sudo dnf install ansible-role-lunasa-hsm
```

2. **Key Manager(barbican)에 대한 configure-barbican.yaml 환경 파일을 생성하고 해당 환경과 관련된 매개변수를 추가합니다.**

```
parameter_defaults:
  BarbicanPkcs11CryptoMKEKLabel: "barbican_mkek_0"
  BarbicanPkcs11CryptoHMACLabel: "barbican_hmac_0"
  BarbicanPkcs11CryptoLogin: "$PKCS_11_USER_PIN"
  BarbicanPkcs11CryptoGlobalDefault: true
  LunasaVars:
    lunasa_client_tarball_name: 610-012382-014_SW_Client_HSM_6.2_RevA.tar.zip
    lunasa_client_tarball_location: https://user:$PASSWORD@http-
server.example.com/luna_software/610-012382-014_SW_Client_HSM_6.2_RevA.tar.zip
    lunasa_client_installer_path: 610-012382-
014_SW_Client_HSM_6.2_RevA/linux/64/install.sh
    lunasa_hsms:
      - hostname: luna-hsm.example.com
        admin_password: "$HSM_ADMIN_PASSWORD"
        partition: myPartition1
        partition_serial: 123456789
```

표 3.3. heat 매개변수

매개변수	현재의
BarbicanSimpleCryptoGlobalDefault	simplecrypto가 글로벌 기본값인지 여부를 결정하는 부울입니다.
BarbicanPkcs11GlobalDefault	PKCS#11이 글로벌 기본값인지 여부를 결정하는 부울입니다.
BarbicanPkcs11CryptoTokenLabel	HSM이 하나 있는 경우 매개 변수의 값은 파티션 레이블입니다. 두 개 이상의 파티션 간에 HA를 사용하는 경우 이 레이블은 HA 그룹에 제공하려는 레이블입니다.

매개변수	현재의
BarbicanPkcs11CryptoLogin	HSM 관리자가 제공하는 HSM에 로그인하는 데 사용되는 PKCS#11 암호입니다.
LunasaVar	
lunasa_client_tarball_name	Luna 소프트웨어 tarball의 이름입니다.
lunasa_client_tarball_location	Luna Software tarball의 HTTPS 서버 위치를 지정하는 URL입니다.
lunasa_client_installer_path	zipped tarball의 install.sh 스크립트 경로입니다.
lunasa_client_rotate_cert	(선택 사항) true로 설정하면 기존 인증서를 교체하기 위해 새 클라이언트 인증서가 생성됩니다. 기본값: false
lunasa_client_working_dir	(선택 사항) 컨트롤러 노드의 작업 디렉터리입니다. 기본값: /tmp/lunasa_client_install
lunasa_hsms	이름, 호스트 이름, admin_password, 파티션 일련 번호를 지정하는 하나 이상의 HSM 목록입니다. 이 목록에 두 개 이상의 HSM을 포함하는 경우 Barbican은 고가용성을 위해 HSM을 구성합니다.

3. **배포 명령에 사용자 지정 configure-barbican.yaml 및 Thales 특정 barbican-backend-pkcs11-llunasa.yaml 환경 파일과 배포와 관련된 기타 템플릿을 포함합니다.**

```
$ openstack overcloud deploy --templates \
  ....
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-pkcs11-lunasa.yaml \
  -e /home/stack/templates/configure-barbican.yaml \
  --log-file overcloud_deployment_with_luna.log
```

3.3. OPENSTACK KEY MANAGER(BARBICAN)와 ENTRUST NSHIELD CONNECT XC HSM 통합

PKCS#11 백엔드를 Entrust nShield Connect XC HSM과 통합하려면 Ansible 역할을 사용하여 컨트롤러에 Entrust 클라이언트 소프트웨어를 다운로드 및 설치하고 사전 정의된 HSM IP 및 인증 정보를 포함하도록 Barbican 구성 파일을 생성합니다.

사전 요구 사항

-

Entrust nShield Connect XC에 대한 벤더 소프트웨어를 제공하는 암호로 보호된 HTTPS 서버입니다.

절차

1. **Barbican에 대한 configure-barbican.yaml 환경 파일을 생성하고 해당 환경과 관련된 매개변수를 추가합니다. 다음 스니펫을 예로 사용합니다.**

```
  parameter_defaults:
    VerifyGlanceSignatures: true
    SwiftEncryptionEnabled: true
    BarbicanPkcs11CryptoLogin: 'sample string'
    BarbicanPkcs11CryptoSlotId: '492971158'
    BarbicanPkcs11CryptoGlobalDefault: true
    BarbicanPkcs11CryptoLibraryPath: '/opt/nfast/toolkits/pkcs11/libcknfast.so'
    BarbicanPkcs11CryptoEncryptionMechanism: 'CKM_AES_CBC'
    BarbicanPkcs11CryptoHMACKeyType: 'CKK_SHA256_HMAC'
    BarbicanPkcs11CryptoHMACKeygenMechanism:
'CKM_NC_SHA256_HMAC_KEY_GEN'
    BarbicanPkcs11CryptoMKEKLabel: 'barbican_mkek_10'
    BarbicanPkcs11CryptoMKEKLength: '32'
    BarbicanPkcs11CryptoHMACLabel: 'barbican_hmac_10'
    BarbicanPkcs11CryptoThalesEnabled: true
    BarbicanPkcs11CryptoEnabled: true
    ThalesVars:
      thales_client_working_dir: /tmp/thales_client_install
      thales_client_tarball_location: https://your server/CipherTools-linux64-dev-12.40.2.tgz
      thales_client_tarball_name: CipherTools-linux64-dev-12.40.2.tgz
      thales_client_path: linux/libc6_11/amd64/nfast
      thales_client_uid: 42481
      thales_client_gid: 42481
      thales_km_data_location: https://your server/kmdata_post_card_creation.tar.gz
      thales_km_data_tarball_name: kmdata_post_card_creation.tar.gz
      thales_rfs_server_ip_address: 192.168.10.12
      thales_hsm_config_location: hsm-C90E-02E0-D947
      nShield_hsms:
        - name: hsm-name.example.com
          ip: 192.168.10.10
      thales_rfs_user: root
      thales_rfs_key: |
        -----BEGIN RSA PRIVATE KEY-----
Sample private key
-----END RSA PRIVATE KEY-----

resource_registry:
  OS::TripleO::Services::BarbicanBackendPkcs11Crypto: /home/stack/tripleo-heat-
templates/puppet/services/barbican-backend-pkcs11-crypto.yaml
```

표 3.4. heat 매개변수

매개변수	현재의
BarbicanSimpleCryptoGlobalDefault	**simplecrypto** 가 글로벌 기본값인지 여부를 결정하는 부울입니다.
BarbicanPkcs11GlobalDefault	**PKCS#11** 이 글로벌 기본값인지 여부를 결정하는 부울입니다.
BarbicanPkcs11CryptoSlotId	Barbican에서 사용할 Virtual HSM의 슬롯 ID입니다.
BarbicanPkcs11CryptoMKEKLabel	이 매개 변수는 HSM에서 생성된 mKEK의 이름을 정의합니다. director는 이 이름을 사용하여 HSM에 이 키를 생성합니다.
BarbicanPkcs11CryptoHMACLabel	이 매개 변수는 HSM에 생성된 HMAC 키의 이름을 정의합니다. director는 이 이름을 사용하여 HSM에 이 키를 생성합니다.
ThalesVars	
thales_client_working_dir	사용자 정의 임시 작업 디렉터리입니다.
thales_client_tarball_location	Entrust 소프트웨어의 HTTPS 서버 위치를 지정하는 URL입니다.
thales_km_data_tarball_name	Entrust 소프트웨어 tarball의 이름입니다.
thales_rfs_key	RFS 서버에 대한 SSH 연결을 가져오는 데 사용되는 개인 키입니다. 이를 RFS 서버에 인증된 키로 추가해야 합니다.

2. **사용자 지정 configure-barbican.yaml 환경 파일과 함께 barbican.yaml 및 Thales 특정 barbican-pkcs11-thales.yaml 환경 파일과 함께 openstack overcloud deploy 명령을 실행할 때 배포에 필요한 기타 템플릿을 포함합니다.**

```
$ 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/containers-prepare-parameter.yaml \
  -e /home/stack/templates/config_lvm.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /home/stack/templates/network/network-environment.yaml \
  -e /home/stack/templates/hostnames.yml \
  -e /home/stack/templates/nodes_data.yaml \
```

```
    -e /home/stack/templates/extra_templates.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-pkcs11-
thales.yaml \
    -e /home/stack/templates/configure-barbican.yaml \
    --log-file overcloud_deployment_with_atos.log
```

검증

1. 테스트 보안을 생성합니다.

```
$ 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 시크릿의 페이로드를 검색합니다.

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

3.3.1. Entrust nShield Connect를 사용한 로드 밸런싱

유효한 **HSM** 배열을 지정하여 **Entrust nShield Connect HSMs**에서 로드 공유를 활성화할 수 있습니다. 두 개 이상의 **HSM**이 나열되면 로드 공유가 활성화됩니다.

이 기능은 이 릴리스에서 기술 프리뷰로 제공되므로 **Red Hat**에서 완전히 지원되지 않습니다. 테스트 용도로만 사용해야 하며 프로덕션 환경에 배포해서는 안 됩니다.

기술 프리뷰 기능에 대한 자세한 내용은 적용 범위 상세 정보를 참조하십시오.

절차

- Entrust nShield Connect HSM에 대한 이름 및 ip 매개변수를 구성할 때 하나 이상의 연결을 지정하면 로드 공유를 활성화합니다.

```
parameter_defaults:
  ....
  ThalesVars:
    ....
    nshield_hsms:
      - name: hsm-name1.example.com
        ip: 192.168.10.10
      - name: hsm-nam2.example.com
        ip: 192.168.10.11
    ....
```

3.4. MKEK 및 HMAC 키 교체

director 업데이트를 사용하여 MKEK 및 HMAC 키를 회전할 수 있습니다.

참고

Barbican의 제한으로 인해 MKEK 및 HMAC는 동일한 키 유형을 갖습니다.

절차

1. 배포 환경 파일에 다음 매개변수를 추가합니다.

```
BarbicanPkcs11CryptoRewrapKeys:          true
```

2. 다음과 유사한 경우 MKEK 및 HMAC 키의 레이블을 변경합니다.

```
    BarbicanPkcs11CryptoMKEKLabel: 'barbican_mkek_10'
    BarbicanPkcs11CryptoHMACLabel: 'barbican_hmac_10'
```

값을 증가하여 라벨을 변경할 수 있습니다.

```
BarbicanPkcs11CryptoMKEKLabel: 'barbican_mkek_11'
BarbicanPkcs11CryptoHMACLabel: 'barbican_hmac_11'
```

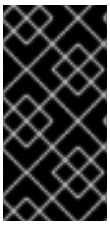
참고

HMAC 키 유형을 변경하지 마십시오.

3. **director를 사용하여 업데이트를 다시 배포하여 업데이트를 적용합니다. director는 MKEK 및 HMAC에 대해 레이블이 지정된 키가 있는지 확인한 다음 생성합니다. 또한 BarbicanPkcs11CryptoRewrapKeys 매개변수를 True 로 설정하면 director는 barbican-manage hsm pkek_rewrap 을 호출하여 기존 pKEK를 모두 다시 래핑합니다.**

4장. OPENSTACK 서비스 암호화 및 검증

barbican을 사용하여 Block Storage(cinder) 암호화 키, 블록 스토리지 볼륨 이미지, Object Storage(swift) 오브젝트 및 Image 서비스(glance) 이미지와 같은 여러 Red Hat OpenStack Platform 서비스를 암호화하고 검증할 수 있습니다.



중요

암호화되지 않은 경우 처음 사용하는 동안 Nova에서 암호화된 볼륨을 포맷합니다. 그런 다음 결과 블록 장치가 컴퓨팅 노드에 표시됩니다.

컨테이너화된 서비스 지침

- 물리적 노드의 호스트 운영 체제에서 찾을 수 있는 구성 파일을 업데이트하지 마십시오(예: /etc/cinder/cinder.conf). 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

  컨테이너를 다시 시작한 후 변경 사항이 적용됩니다.

4.1. OBJECT STORAGE(SWIFT) AT-REST 오브젝트 암호화

기본적으로 **Object Storage(swift)**에 업로드된 오브젝트는 암호화되지 않은 상태로 저장됩니다. 이로 인해 파일 시스템에서 직접 오브젝트에 액세스할 수 있습니다. 디스크가 삭제되기 전에 제대로 삭제되지 않은 경우 보안 위험이 발생할 수 있습니다. **barbican**을 활성화하면 **Object Storage** 서비스**(swift)**에서 저장된 오브젝트를 투명하게 암호화하고 암호 해독할 수 있습니다. **At-rest** 암호화는 디스크에 저장되는 동안 암호화되는 개체를 참조하는 점에서 전송 중 암호화와 다릅니다.

Swift는 **swift**에 업로드할 때 오브젝트가 자동으로 암호화되고 사용자에게 제공되면 자동으로 암호 해독되므로 이러한 암호화 작업을 투명하게 수행합니다. 이 암호화 및 암호 해독은 **barbican**에 저장된 동일한 **(symmetric)** 키를 사용하여 수행됩니다.

참고

데이터가 현재 암호화된 상태로 저장되므로 암호화 및 **swift** 클러스터에 데이터를 추가한 후에는 암호화를 비활성화할 수 없습니다. 결과적으로 동일한 키로 암호화를 다시 활성화할 때까지 암호화가 비활성화된 경우 데이터를 읽을 수 없습니다.

사전 요구 사항

- **OpenStack Key Manager**가 설치 및 활성화됨

프로세스

1. 환경 파일에 **SwiftEncryptionEnabled: True** 매개변수를 포함하는 다음 **/home/stack/overcloud_deploy.sh** 를 사용하여 **openstack overcloud deploy** 를 다시 실행합니다.

2. **swift**가 **at-rest** 암호화를 사용하도록 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
```

결과에 암호화에 대한 항목이 포함되어야 합니다.

4.2. BLOCK STORAGE(CINDER) 볼륨 암호화

barbican을 사용하여 **Block Storage(cinder)** 암호화 키를 관리할 수 있습니다. 이 구성은 **LUKS**를 사용하여 부팅 디스크를 포함하여 인스턴스에 연결된 디스크를 암호화합니다. 키 관리는 사용자에게 투명합니다. **luks** 를 암호화 유형으로 사용하여 새 볼륨을 생성할 때 **cinder**는 볼륨에 대한 대칭 키 시크릿을 생성하여 **barbican**에 저장합니다. 인스턴스를 부팅하거나 암호화된 볼륨을 연결할 때 **nova**는 **barbican**에서 키를 검색하고 로컬로 컴퓨팅 노드에 **Libvirt** 시크릿으로 시크릿을 저장합니다.

절차

1. **cinder-volume** 및 **nova-compute** 서비스를 실행하는 노드에서 **nova** 및 **cinder**가 모두 키 관리에 **barbican**을 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 생성할 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.

```
$ 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     |
+--------------------+--------------------------------------+
```

생성된 시크릿은 barbican 백엔드에 자동으로 업로드됩니다.

참고

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

4.

barbican 시크릿 UUID를 가져옵니다. 이 값은 encryption_key_id 필드에 표시됩니다.

```
$ cinder --os-volume-api-version 3.64 volume show Encrypted-Test-Volume
+------------------------------+--------------------------------------+
|Property                      |Value                                 |
+------------------------------+--------------------------------------+
|attached_servers              |[]                                    |
|attachment_ids                |[]                                    |
|availability_zone             |nova                                  |
|bootable                      |false                                 |
|cluster_name                  |None                                  |
|consistencygroup_id           |None                                  |
|created_at                    |2022-07-28T17:35:26.000000            |
|description                   |None                                  |
|encrypted                     |True                                  |
```

```
|encryption_key_id            |0944b8a8-de09-4413-b2ed-38f6c4591dd4 |
|group_id                     |None                                 |
|id                           |a0b51b97-0392-460a-abfa-093022a120f3 |
|metadata                     |                                     |
|migration_status             |None                                 |
|multiattach                  |False                                |
|name                         |vol                                  |
|os-vol-host-attr:host        |hostgroup@tripleo_iscsi#tripleo_iscsi|
|os-vol-mig-status-attr:migstat|None                                |
|os-vol-mig-status-attr:name_id|None                                |
|os-vol-tenant-attr:tenant_id  |a2071ece39b3440aa82395ff7707996f     |
|provider_id                  |None                                 |
|replication_status           |None                                 |
|service_uuid                 |471f0805-072e-4256-b447-c7dd10ceb807 |
|shared_targets               |False                                |
|size                         |1                                    |
|snapshot_id                  |None                                 |
|source_volid                 |None                                 |
|status                       |available                            |
|updated_at                   |2022-07-28T17:35:26.000000           |
|user_id                      |ba311b5c2b8e438c951d1137333669d4     |
|volume_type                  |LUKS                                 |
|volume_type_id               |cc188ace-f73d-4af5-bf5a-d70ccc5a401c |
+-----------------------------+-------------------------------------+
```

참고

encryption_key_id 값을 표시하려면 Cinder CLI와 함께 --os-volume-api-version 3.64 매개변수를 사용해야 합니다. 동등한 OpenStack CLI 명령이 없습니다.

5. **barbican을 사용하여 디스크 암호화 키가 있는지 확인합니다. 이 예에서 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/0944b8a8-de09-4413-b2ed-38f6c4591dd4 | None |
2018-01-22T02:23:15+00:00 | ACTIVE | {u'default': u'application/octet-stream'} | aes       |
256 | symmetric   | None | None       |
+------------------------------------------------------------------------------------+------+------------------
---------+--------+-----------------------------------------+----------+-----------+------------+------
+------------+
```

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

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

그러면 볼륨이 게스트 운영 체제에 표시되고 기본 제공 툴을 사용하여 마운트할 수 있습니다.

4.2.1. 블록 스토리지 볼륨을 OpenStack Key Manager로 마이그레이션

이전에 **ConfKeyManager** 를 사용하여 디스크 암호화 키를 관리한 경우 **barbican**으로 마이그레이션하기 위해 **scope** 내의 **encryption_key_id** 항목에 대한 데이터베이스를 스캔하여 볼륨을 **OpenStack Key Manager**로 마이그레이션할 수 있습니다. 각 항목은 새 **barbican** 키 **ID**를 가져오고 기존 **ConfKeyManager** 시크릿은 유지됩니다.

참고

- 이전에는 **ConfKeyManager** 를 사용하여 암호화된 볼륨에 대한 소유권을 다시 할당할 수 있었습니다. **barbican**에서 관리하는 키가 있는 볼륨에는 이 작업을 수행할 수 없습니다.
- **barbican**을 활성화하면 기존 키**mgr** 볼륨이 손상되지 않습니다.

사전 요구 사항

마이그레이션하기 전에 **Barbican**에서 관리하는 암호화된 볼륨과 **ConfKeyManager** 를 사용하는 볼륨 간의 다음 차이점을 검토하십시오.

- 현재 **barbican** 시크릿의 소유권을 이전할 수 없기 때문에 암호화된 볼륨의 소유권을 이전할 수 없습니다.
- **Barbican**은 보안을 읽고 삭제할 수 있는 사람을 제한적으로 일부 **cinder** 볼륨 작업에 영향을 줄 수 있습니다. 예를 들어 사용자는 다른 사용자의 볼륨을 연결, 분리 또는 삭제할 수 없습니다.

절차

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. **cinder-volume** 및 **cinder-backup** 서비스는 마이그레이션 프로세스를 자동으로 시작합니다. 로그 파일을 확인하여 마이그레이션에 대한 상태 정보를 볼 수 있습니다.

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

4. 마이그레이션이 완료되었음을 나타내는 메시지의 로그를 모니터링하고 **ConfKeyManager all-zeros** 암호화 키 **ID**를 사용하여 더 이상 볼륨이 없는지 확인합니다.

5. **cinder.conf** 및 **nova.conf** 에서 **fixed_key** 옵션을 제거합니다. 이 설정이 구성된 노드를 결정해야 합니다.

6. **cinder** 서비스에서 작성자 역할을 제거합니다.

검증

- 프로세스를 시작하면 이러한 항목 중 하나가 로그 파일에 표시됩니다. 이는 마이그레이션이 올바르게 시작되었는지 여부를 나타냅니다. 그렇지 않으면 마이그레이션이 발생한 문제를 식별합니다.
  - **ConfKeyManager**가 여전히 사용 중이므로 암호화 키를 마이그레이션하지 않습니다.
  - **ConfKeyManager**의 **fixed_key**가 사용되지 않기 때문에 암호화 키를 마이그레이션하지 않습니다.
  - **'XXX' key_manager** 백엔드로 마이그레이션하기 때문에 암호화 키를 마이그레이션하지 않습니다. - 이 메시지는 나타나지 않을 수 있습니다. 이는 **barbican** 이외의 다른 **Key Manager** 백엔드가 발생하는 코드를 처리하는 안전 검사입니다. 이는 코드에서 하나의 마이

그레이션 시나리오만 지원하기 때문입니다. **ConfKeyManager** *에서* **barbican** *까지의* 마이그레이션 시나리오만 지원되기 때문입니다.

- 이 호스트와 연결된 볼륨이 없기 때문에 암호화 키를 마이그레이션하지 않습니다. **-cinder-volume** 이 여러 호스트에서 실행되고 특정 호스트에 연결된 볼륨이 없을 때 발생할 수 있습니다. 이는 모든 호스트가 자체 볼륨을 처리할 책임이 있기 때문에 발생합니다.

- **ConfKeyManager** 키의 마이그레이션 시작

- 볼륨 **<UUID>** 암호화 키를 **Barbican**로 마이그레이션하면 마이그레이션 중에 호스트의 모든 볼륨을 검사하고 볼륨이 계속 **ConfKeyManager**의 키 **ID**를 사용하는 경우(모든 **0 0000-0000-0000-00000000**)) 이 메시지가 나타납니다.

  - **cinder-backup** 의 경우 이 메시지는 약간 다른 대문자를 사용합니다: 볼륨 마이그레이션 [또는 백업 마이그레이션]

- 각 호스트가 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.

  - **ConfKeyManager**의 **all-zeros** 암호화 키 **ID**를 사용하는 여전히 **%d** 볼륨이 있습니다.

  - **ConfKeyManager**의 **all-zeros** 암호화 키 **ID**를 사용하는 여전히 **%d** 백업이 있습니다.

  이러한 메시지는 **cinder-volume** 및 **cinder-backup** 로그에 표시될 수 있습니다. 각 서비스는 자체 항목 마이그레이션만 처리하지만 서비스는 다른 서비스의 상태를 알고 있습니다. 결과적으로 **cinder-backup** 에 마이그레이션할 백업이 있는지 여부를 **cinder-volume** 에서 알고 있으며 **cinder-backup** 에는 **cinder-volume** 서비스에 마이그레이션할 볼륨이 있는지 확인합니다.

각 호스트는 자체 볼륨만 마이그레이션하지만 요약 메시지는 마이그레이션이 여전히 필요한지 여부에 대한 글로벌 평가를 기반으로 하므로 모든 볼륨의 마이그레이션이 완료되었는지 확인할 수 있습니다.

cleanup

키 **ID**를 **barbican**로 마이그레이션한 후 고정 키는 구성 파일에 남아 있습니다. **fixed_key** 값은 **.conf** 파일에서 암호화되지 않기 때문에 일부 사용자에게 보안 문제가 표시될 수 있습니다.

이 문제를 해결하려면 **nova** 및 **cinder** 구성에서 **fixed_key** 값을 수동으로 제거할 수 있습니다. 그러나 이 값을 계속 사용하는 디스크에 액세스할 수 없기 때문에 계속하기 전에 먼저 테스트한 후 로그 파일의 출력을 검토합니다.

중요

encryption_key_id 는 최근에 **Backup** 테이블에만 추가되었습니다. 결과적으로 암호화된 볼륨의 기존 백업이 존재할 수 있습니다. **all-zeros encryption_key_id** 는 백업 자체에 저장되지만 **Backup** 데이터베이스에는 표시되지 않습니다. 따라서 마이그레이션 프로세스에서 여전히 **all-zeros ConfKeyMgr** 키 **ID**에 의존하는 암호화된 볼륨의 백업이 존재하는지 여부를 확인하는 것은 불가능합니다.

1. 기존 **fixed_key** 값을 검토합니다. 두 서비스 모두에 대해 값이 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
```

중요

기존 **fixed_key** 값의 백업을 만듭니다. 이렇게 하면 문제가 발생하거나 이전 암호화 키를 사용하는 백업을 복원해야 하는 경우 값을 복원할 수 있습니다.

2. **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
```

문제 해결

barbican 시크릿은 요청자에게 작성자 역할이 있는 경우에만 생성할 수 있습니다. 즉 **cinder** 서비스 자체에는 작성자 역할이 필요하며, 그렇지 않으면 다음과 유사한 로그 시퀀스가 발생합니다.

1. ConfKeyManager 키의 마이그레이션 시작

2. 볼륨 <UUID> 암호화 키를 Barbican으로 마이그레이션

3. 암호화 키를 마이그레이션하는 동안 오류 발생: 금지: 시크릿 생성 시도가 허용되지 않음 - 사용자/프로젝트 권한을 검토하십시오.

4. ConfKeyManager의 all-zeros 암호화 키 ID를 사용하는 여전히 %d 볼륨이 있습니다.

핵심 메시지는 세 번째 메시지입니다: 시크릿 생성 시도는 허용되지 않습니다. 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.

결과적으로 마이그레이션 시 다음 시도에 성공해야 합니다.

4.3. BLOCK STORAGE(CINDER) 볼륨 이미지 검증

Block Storage 서비스(cinder)는 이미지 생성에서 볼륨을 생성하는 동안 다운로드되고 서명된 이미지의 서명을 자동으로 검증합니다. 이미지가 볼륨에 기록되기 전에 서명이 검증됩니다. 성능을 개선하기 위해 Block Storage Image-Volume 캐시를 사용하여 새 볼륨을 생성하기 위해 검증된 이미지를 저장할 수 있습니다.

참고

Red Hat Ceph Storage 또는 RBD 볼륨에서 Cinder 이미지 서명 검증은 지원되지 않습니다.

절차

1. 컨트롤러 노드에 로그인합니다.

2. 다음 옵션 중 하나를 선택합니다.

- 볼륨 로그 **/var/log/containers/cinder/cinder-volume.log** 에서 **cinder**의 이미지 검증 활동을 확인합니다.

  예를 들어 인스턴스가 부팅될 때 다음 항목을 기대할 수 있습니다.

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

- **openstack volume list** 및 **cinder volume show** 명령을 사용합니다.

  a. **openstack volume list** 명령을 사용하여 볼륨 **ID**를 찾습니다.

  b. 컴퓨팅 노드에서 **cinder volume show** 명령을 실행합니다.

  ```
  cinder volume show <VOLUME_ID>
  ```

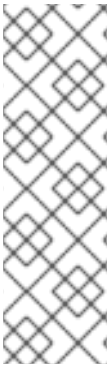
3. 확인된 행 서명 **: True** 를 사용하여 **volume_image_metadata** 섹션을 찾습니다.

```
$ cinder show d0db26bb-449d-4111-a59a-6fbb080bb483
+--------------------------------+------------------------------------------------+
| Property                       | Value                                          |
+--------------------------------+------------------------------------------------+
| attached_servers               | []                                             |
| attachment_ids                 | []                                             |
| availability_zone              | nova                                           |
| bootable                       | true                                           |
| consistencygroup_id            | None                                           |
| created_at                     | 2018-10-12T19:04:41.000000                     |
| description                    | None                                           |
| encrypted                      | True                                           |
| id                             | d0db26bb-449d-4111-a59a-6fbb080bb483           |
| metadata                       |                                                |
| migration_status               | None                                           |
| multiattach                    | False                                          |
| name                           | None                                           |
| os-vol-host-attr:host          | centstack.localdomain@nfs#nfs                  |
| os-vol-mig-status-attr:migstat | None                                           |
| os-vol-mig-status-attr:name_id | None                                           |
```

```
| os-vol-tenant-attr:tenant_id   | 1a081dd2505547f5a8bb1a230f2295f4                |
| replication_status             | None                                            |
| size                           | 1                                               |
| snapshot_id                    | None                                            |
| source_volid                   | None                                            |
| status                         | available                                       |
| updated_at                     | 2018-10-12T19:05:13.000000                      |
| user_id                        | ad9fe430b3a6416f908c79e4de3bfa98                |
| volume_image_metadata          | checksum : f8ab98ff5e73ebab884d80c9dc9c7290     |
|                                | container_format : bare                         |
|                                | disk_format : qcow2                             |
|                                | image_id : 154d4d4b-12bf-41dc-b7c4-35e5a6a3482a |
|                                | image_name : cirros-0.3.5-x86_64-disk           |
|                                | min_disk : 0                                    |
|                                | min_ram : 0                                     |
|                                | signature_verified : False                      |
|                                | size : 13267968                                 |
| volume_type                    | nfs                                             |
+--------------------------------+-------------------------------------------------+
```

참고

스냅샷은 **Image** 서비스(**glance**) 이미지로 저장됩니다. 서명된 이미지를 확인하도록 **Compute** 서비스(**nova**)를 구성하는 경우 **Glance**에서 이미지를 수동으로 다운로드하고 이미지에 서명한 다음 이미지를 다시 업로드해야 합니다. 이는 스냅샷이 서명된 이미지로 생성된 인스턴스인지 아니면 서명된 이미지에서 생성된 볼륨에서 부팅되었는지 여부입니다.



참고

볼륨은 **Image** 서비스(**glance**) 이미지로 업로드할 수 있습니다. 원본 볼륨을 부팅할 수 있는 경우 이미지를 사용하여 **Block Storage** 서비스(**cinder**)에 부팅 가능한 볼륨을 생성할 수 있습니다. 서명된 이미지를 확인하도록 **Block Storage** 서비스를 구성한 경우 **Glance**에서 이미지를 수동으로 다운로드하고 이미지를 사용하기 전에 이미지 서명을 계산하고 모든 적절한 이미지 서명 속성을 업데이트해야 합니다. 자세한 내용은 **4.5절. "스냅샷 검증"**의 내용을 참조하십시오.

추가 리소스

- **Block Storage 서비스(cinder) 구성**

4.3.1. 볼륨 이미지 암호화 키 자동 삭제

Block Storage 서비스(**cinder**)는 암호화된 볼륨을 **Image** 서비스(**glance**)에 업로드할 때 키 관리 서비스(**barbican**)에 암호화 키를 생성합니다. 이렇게 하면 암호화 키와 저장된 이미지 간에 **1:1** 관계가 생성됩니다.

암호화 키를 삭제하면 키 관리 서비스의 리소스가 무제한으로 사용되지 않습니다. 블록 스토리지, 키 관리 및 이미지 서비스는 키 삭제를 포함하여 암호화된 볼륨의 키를 자동으로 관리합니다.

블록 스토리지 서비스는 볼륨 이미지에 두 개의 속성을 자동으로 추가합니다.

- **cinder_encryption_key_id -** 키 관리 서비스에서 특정 이미지에 저장하는 암호화 키의 식별자입니다.
- **cinder_encryption_key_deletion_policy -** 이미지 서비스에 이 이미지와 연결된 키를 삭제할지 여부를 알리는 정책입니다.

중요

이러한 속성의 값은 자동으로 할당됩니다. 의도하지 않은 데이터 손실을 방지하려면 이러한 값을 조정하지 마십시오.

볼륨 이미지를 생성할 때 **Block Storage** 서비스는 **cinder_encryption_key_deletion_policy** 속성을 **on_image_deletion** 으로 설정합니다. 볼륨 이미지를 삭제하면 **cinder_encryption_key_deletion_policy** 가 **on_image_deletion** 과 같은 경우 이미지 서비스에서 해당 암호화 키를 삭제합니다.

중요

Red Hat은 **cinder_encryption_key_id** 또는 **cinder_encryption_key_deletion_policy** 속성을 수동으로 조작하는 것을 권장하지 않습니다. 다른 목적으로 **cinder_encryption_key_id** 값으로 식별되는 암호화 키를 사용하는 경우 데이터 손실 위험이 있습니다.

4.4. IMAGE 서비스(GLANCE) 이미지에 서명

업로드된 이미지가 변조되지 않았는지 확인하도록 **Image** 서비스(**glance**)를 구성할 때 해당 이미지를 사용하여 인스턴스를 시작하기 전에 이미지에 서명해야 합니다. **openssl** 명령을 사용하여 **barbican**에 저장된 키로 이미지에 서명한 다음 관련 서명 정보를 사용하여 이미지를 **Glance**에 업로드합니다. 결과적으로 매번 사용하기 전에 이미지의 서명을 확인하고 서명이 일치하지 않으면 인스턴스 빌드 프로세스가 실패합니다.

사전 요구 사항

- **OpenStack Key Manager**가 설치 및 활성화됨

프로세스

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

2. **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 **Image** 및 **Compute** 서비스의 백엔드로 사용하면 **CoW** 복제가 생성됩니다. 따라서 이미지 서명 확인을 수행할 수 없습니다.

3. **glance**가 **barbican**을 사용하도록 구성되어 있는지 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
```

4. 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
```

5. **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 결과 **UUID**를 기록합니다. 이 예에서 인증서의 **UUID**는 **5df14c2b-f221-4a02-948e-48a61ed3f5b** 입니다.

6. **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
```

7. 생성된 **.signature** 파일을 ***base64*** 형식으로 변환합니다.

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

8. ***base64*** 값을 변수에 로드하여 후속 명령에서 사용합니다.

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

9. 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                        | None                                                                         |
| container_format                | bare                                                                         |
| created_at                      | 2018-01-23T05:37:31Z                                                         |
| disk_format                     | qcow2                                                                        |
| id                              | d3396fa0-2ea2-4832-8a77-d36fa3f2ab27                                         |
| img_signature                   |
lcI7nGgoKxnCyOcsJ4abbEZEpzXByFPIgiPeiT+Otjz0yvW00KNN3fI0AA6tn9EXrp7fb2xBDE4UaO3v |
|                                 |
lFquV/s3mU4LcCiGdBAl3pGsMlmZZIQFVNcUPOaayS1kQYKY7kxYmU9iq/AZYyPw37KQI52s
```

```
mC/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                         | queued                                                                           |
| tags                           | []                                                                               |
| updated_at                     | 2018-01-23T05:37:31Z                                                             |
| virtual_size                   | None                                                                             |
| visibility                     | shared                                                                           |
+--------------------------------+----------------------------------------------------------------------------------
----+
```

10. **Compute log: /var/log/containers/nova/nova-compute.log** 에서 **Glance**의 이미지 검증 활동을 볼 수 있습니다. 예를 들어 인스턴스가 부팅될 때 다음 항목을 기대할 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
```

4.5. 스냅샷 검증

스냅샷은 **Image** 서비스**(glance)** 이미지로 저장됩니다. 서명된 이미지를 확인하도록 **Compute** 서비스**(nova)**를 구성하는 경우 서명된 이미지가 있는 인스턴스에서 생성된 경우에도 스냅샷은 서명해야 합니다.

절차

1. **Glance**에서 스냅샷 다운로드

```
openstack image save --file <local-file-name> <image-name>
```

2. 스냅샷을 검증하기 위해 서명을 생성합니다. 이 프로세스는 서명을 생성하여 이미지의 유효성을 검사할 때 사용하는 프로세스와 동일합니다. 자세한 내용은 **Image** 서비스**(glance)** 이미지 검증을 참조하십시오.

3. **이미지 속성을 업데이트합니다.**

```
openstack image set \
  --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" \
  <image_id_of_the_snapshot>
```

4. **선택 사항: 파일 시스템에서 다운로드한 Glance 이미지를 제거합니다.**

```
rm <local-file-name>
```