Red Hat OpenStack Platform 13

보안 및 강화 가이드

모범 사례, 규정 준수 및 보안 강화

Last Updated: 2023-05-25

Red Hat OpenStack Platform 13 보안 및 강화 가이드

모범 사례, 규정 준수 및 보안 강화

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

본 가이드는 Red Hat OpenStack Platform 환경의 보안 강화에 관한 모범 사례 조언 및 개념 정보를 제공합니다.

차례

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 이를 개선하는지 알려주십시오.

DDF(직접 문서 피드백) 기능 사용

특정 문장, 단락 또는 코드 블록에 대한 직접 주석은 **피드백 추가** DDF 기능을 사용하십시오.

1. *다중 페이지 HTML* 형식으로 설명서를 봅니다.
2. 문서 오른쪽 상단에 **Feedback** (피드백) 버튼이 표시되는지 확인합니다.
3. 주석 처리하려는 텍스트 부분을 강조 표시합니다.
4. **피드백 추가**를 클릭합니다.
5. 주석을 사용하여 **Add Feedback** (피드백 추가) 필드를 작성합니다.
6. 선택 사항: 설명서 팀이 문제에 대한 자세한 내용을 문의할 수 있도록 이메일 주소를 추가하십시오.
7. **Submit(제출)**을 클릭합니다.

1장. 보안 소개

RHOSP(Red Hat Openstack Platform)와 함께 제공되는 툴을 사용하여 계획 및 운영 시 보안 우선 순위를 지정하면 사용자의 개인 정보 보호 및 데이터 보안에 대한 기대를 충족할 수 있습니다. 보안 표준을 구현하지 않으면 가동 중지 시간이나 데이터 침해가 발생할 수 있습니다. 사용 사례는 감사 및 규정 준수 프로세스를 통과해야 하는 법률의 적용을 받을 수 있습니다. 참고

이 가이드의 지침에 따라 사용자 환경의 보안을 강화합니다. 그러나 이러한 권장 사항은 보안 또는 규정 준수를 보장하지 않습니다. 환경의 고유한 요구 사항으로 인해 보안을 평가해야 합니다.

Ceph 강화에 대한 자세한 내용은 데이터 보안 및 강화 가이드를 참조하십시오.

1.1. RED HAT OPENSTACK PLATFORM SECURTIY

기본적으로 RHOSP(Red Hat OpenStack Platform) director는 보안을 위한 다음 툴 및 액세스 제어를 사용하여 오버클라우드를 생성합니다.

SElinux

SELinux는 각 프로세스가 모든 작업에 대해 명시적 권한이 있어야 하는 액세스 제어를 제공하여 RHOSP의 보안 강화를 제공합니다.

Podman

Podman을 컨테이너 툴로 사용하는 것은 루트 액세스 권한이 있는 프로세스가 기능해야 하는 클라이언트/서버 모델을 사용하지 않으므로 RHOSP의 안전한 옵션입니다.

시스템 액세스 제한

오버클라우드 배포 중에 director가 heat-admin을 위해 생성하는 SSH 키 또는 오버클라우드에서 생성한 SSH 키를 사용하여 오버클라우드 노드에만 로그인할 수 있습니다. 암호와 함께 SSH를 사용하여 오버클라우드 노드에 로그인하거나 root를 사용하여 오버클라우드 노드에 로그인할 수 없습니다.

조직의 요구 사항 및 신뢰 수준에 따라 다음 추가 보안 기능을 사용하여 director를 구성할 수 있습니다.

- 공용 TLS 및 TLS-everywhere
- OpenStack Key Manager와 하드웨어 보안 모듈 통합(barbican)
- 서명된 이미지 및 암호화된 볼륨
- 워크플로우 실행을 사용한 암호 및 fernet 키 순환

1.2. 보안 영역

보안 영역은 일반적인 보안 문제를 공유하는 일반적인 리소스, 애플리케이션, 네트워크 및 서버입니다. 보안 영역은 동일한 인증 및 권한 부여 요구 사항 및 사용자를 공유해야 합니다.

클라우드의 고유한 아키텍처, 환경에 대한 허용 가능한 신뢰 수준 및 표준화된 요구 사항에 따라 자체 보안 영역을 재정의해야 할 수도 있습니다. 영역과 신뢰 요구 사항은 클라우드 인스턴스가 퍼블릭, 프라이빗 또는 하이브리드인지 여부에 따라 다를 수 있습니다.

가장 신뢰받는 OpenStack 클라우드를 배포하는 데 필요한 최소 영역은 다음과 같습니다.

- **퍼블릭 영역**: 공용 영역은 인스턴스의 외부 연결을 위해 유동 IP 및 SNAT와 같은 공용 방향 API 및 neutron 외부 네트워크를 호스팅합니다. 이 영역은 신뢰할 수 없는 클라우드 인프라 영역입니다. 공용 또는 해당 지역 외의 네트워크에 의한 클라우드 액세스 또는 Red Hat OpenStack Platform 배포 외부의 책임을 가리킵니다.

이 영역을 통과하는 기밀성 또는 무결성 요구 사항이 있는 모든 데이터는 보상 제어를 사용하여 보호해야 합니다.

- **게스트 영역**: 게스트 영역은 VXLAN 또는 GENEVE에 관계없이 프로젝트 네트워크를 호스팅합니다. 인스턴스 사용 시 엄격한 제어 권한이 없거나 인스턴스에 대한 무제한 인터넷 액세스를 허용하는 공용 및 프라이빗 클라우드 공급업체에 대해 신뢰할 수 없습니다.

사설 클라우드 공급자인 경우 기본 하드웨어 및 물리적 네트워크를 포함하도록 제어가 구현되어 기본 하드웨어 및 물리적 네트워크를 포함할 수 있는 경우 이 네트워크를 내부 및 신뢰할 수 있는 것으로 간주할 수 있습니다.

- **스토리지 액세스 영역**: 스토리지 액세스 영역은 스토리지 관리, 모니터링 및 클러스터링은 물론 스토리지 트래픽 자체를 위한 것입니다.

이 네트워크를 통해 전송된 데이터를 사용하려면 높은 수준의 무결성과 기밀성이 필요합니다. 또한 강력한 가용성 요구 사항이 있을 수 있습니다.

복제 요구 사항을 제외하고, 이 영역에 배포된 스토리지 어플라이언스 구성 요소 이외의 클라우드 외부에서 이 네트워크에 액세스하도록 하지 마십시오. 보안 관점에서는 민감합니다.

- **관리 영역**: 관리 영역에는 언더클라우드, 호스트 운영 체제, 서버 하드웨어, 물리적 네트워킹 및 Red Hat OpenStack Platform director 컨트롤 플레인이 포함됩니다.

관리 네트워크는 시스템 관리, 모니터링 또는 백업에 사용됩니다. 이 영역에서 OpenStack API 또는 제어 인터페이스를 호스팅하지 마십시오.

- **관리 영역**: admin 영역은 시스템 관리, 모니터링 또는 백업에 대한 트래픽을 호스팅하지만 OpenStack API 또는 제어 인터페이스가 호스팅되지 않는 곳입니다.

1.3. 보안 영역 연결

신뢰 수준 또는 인증 요구 사항이 서로 다른 여러 보안 영역에 걸쳐 있는 구성 요소를 신중하게 구성해야 합니다. 이러한 연결은 종종 네트워크 아키텍처의 약점이 되며 연결된 모든 영역의 신뢰 수준이 가장 높은 보안 요구 사항을 충족하도록 구성해야 합니다. 대부분의 경우 보안 제어는 연결된 영역을 제어하는 것이 공격 가능성으로 인해 주요 우려 사항입니다. 영역을 충족하는 지점은 추가 공격 서비스를 제공하고 공격자가 더욱 민감한 배포 부분으로 공격을 마이그레이션할 수 있는 기회를 추가합니다.

OpenStack 운영자는 경우에 따라 OpenStack 운영자가 상주하는 영역보다 높은 표준의 통합 지점 보안을 고려할 수 있습니다. API 엔드포인트의 위의 예에서, 상대 팀은 이러한 영역이 완전히 격리되지 않은 경우 관리 영역 내의 내부 또는 관리 API에 액세스하거나 액세스하기 위해 퍼블릭 영역의 공용 API 엔드포인트를 대상으로 잠재적으로 퍼블릭 API 엔드포인트를 대상으로 할 수 있었습니다.

OpenStack의 설계는 보안 영역의 분리가 어렵습니다. 일반적으로 핵심 서비스는 두 개 이상의 영역에 걸쳐 있으므로 보안 제어를 적용할 때 특히 고려해야 합니다.

1.4. 위협 분류, 행위자 및 공격 벡터

대부분의 유형의 클라우드 배포, 공용, 프라이빗 또는 하이브리드는 일부 형태의 공격에 노출됩니다. 이 섹션에서는 공격자를 분류하고 각 보안 영역에서 잠재적인 공격 유형을 요약합니다.

1.4.1. 위협 행위자

위협 행위자는 방어하기 위해 시도할 수 있는 변위 클래스를 참조할 수 있는 추상적인 방법입니다. 행위자가 보다 능숙할수록 성공적인 공격 완화 및 예방에 필요한 보안 제어가 더욱 엄격합니다. 보안은 요구 사항에 따라 편의성, 방어 및 비용 조정의 문제입니다. 경우에 따라 여기에 설명된 모든 위협 행위자에 대해 클

라우드 배포를 보호할 수 없습니다. OpenStack 클라우드를 배포할 때는 배포 및 사용에 대한 균형이 어느 정도인지 결정해야 합니다.

위험 평가의 일환으로 저장하는 데이터 유형과 액세스 가능한 리소스도 고려해야 합니다. 이는 특정 행위자에게도 영향을 미치기 때문입니다. 그러나 데이터가 위협 행위자들을 선호하지 않더라도, 예를 들어, botnet에 참여하거나 권한 없는 채광을 실행하는 등 컴퓨팅 리소스에 관심을 가질 수 있습니다.

- 국가-State Actors - 이것이 가장 능력 있는 대담입니다. 국가 상태 행위자는 대상에 대해 엄청난 리소스를 가져올 수 있습니다. 다른 행위자 이외의 능력을 보유하고 있습니다. 개인과 기술 모두에 대해 믿을 수 없는 엄격한 통제 없이 이러한 행위자를 방어하는 것은 매우 어렵습니다.

- 심각한 조직 오류 - 이 클래스는 고도로 성능 및 재정적으로 구동되는 공격자 그룹을 설명합니다. 이들은 사내 위협 개발 및 목표 연구 자금을 지원할 수 있습니다. 최근 몇 년 동안 사이버 대기업인 러시아어 비즈니스 네트워크와 같은 조직의 증가는 사이버 공격이 어떻게 상품화되었는지를 보여 주었습니다. 사이버 보안은 심각한 조직 내에 있으며, 심각한 조직 내에 있습니다.

- 높은 지원 그룹 - 이는 일반적으로 상업적으로 자금을 투입하지 않지만 서비스 공급자와 클라우드 운영자에게 심각한 위협을 줄 수 있는 'Hacktivist' 유형 조직을 의미합니다.

- 동기부여하는 개인 연습만으로 이들 공격자는 악의적인 직원 또는 악의적인 직원, 영향을 받는 고객, 소규모 산업 스스피션 등 여러 신장에 도착하게 됩니다.

- 스크립트 Kiddies - 이러한 공격자는 특정 조직을 대상으로 하지 않지만 자동 취약점 검사 및 악용을 실행합니다. 종종 이러한 행위자 중 한 명으로 인해 문제가 발생하는 것은 조직의 평판에 큰 위험이 있습니다.

다음 사례는 위에서 식별한 몇 가지 위험을 완화하는 데 도움이 될 수 있습니다.

- 보안 업데이트 - 네트워킹, 스토리지 및 서버 하드웨어를 포함하여 기본 물리적 인프라의 포괄적인 보안 포스처를 고려해야 합니다. 이러한 시스템에는 자체 보안 강화 방법이 필요합니다. Red Hat OpenStack Platform 배포를 위해 보안 업데이트를 정기적으로 테스트하고 배포할 계획입니다.

- 액세스 관리 - 개인에게 시스템 액세스 권한을 부여할 때는 *최소 권한의 원칙을* 적용하고 실제로 필요한 세분화된 시스템 권한만 부여해야 합니다. AAA 연습(액세스, 권한 부여 및 회계)을 사용하여 이 정책을 시행할 수 있습니다. 이 접근 방식은 또한 시스템 관리자로부터 악의적인 행위자 및 오타 오류의 위험을 완화하는 데 도움이 될 수 있습니다.

- 내부자 관리 - 역할 기반 액세스 제어(최소 액세스 필요), 내부 인터페이스에서 암호화를 사용하여 인증/인증/인증 보안을 사용하여 악의적인 내부자의 위협을 완화할 수 있습니다(예: 중앙 집중식 ID 관리). 직무 분리 및 비정규직 역할 순환과 같은 추가적인 비기술 옵션을 고려할 수도 있습니다.

1.4.2. 아웃바운드 공격 및 평가 위험

클라우드 배포로 인한 아웃바운드 오용에 대해 신중하게 고려해야 합니다. 클라우드 배포는 많은 리소스를 사용할 수 있는 경향이 있습니다. 악의적인 직원 같은 해커는 이러한 리소스를 악용하여 악의적인 목적으로 사용할 수 있습니다. 계산 서비스를 사용하는 클라우드는 이상적이고 무차별 엔진에 사용됩니다. 이 문제는 사용자가 대략적으로 계산할 수 없기 때문에 공용 클라우드에 대해 누르는 것이며 아웃바운드 공격에 대해 수많은 일회용 인스턴스를 빠르게 실행할 수 있습니다. 방지 방법에는 송신 보안 그룹, 트래픽 검사, 침입 탐지 시스템, 고객 교육 및 인식, 사기 및 오용 완화 전략이 포함됩니다. 또는 에 의해 또는 인터넷과 같은 공용 네트워크에 액세스하여 액세스할 수 있는 배포의 경우 프로세스 및 인프라가 이상적으로 탐지되고 아웃바운드 오용을 해결하기 위해 이루어져야 합니다.

1.5. 지원 소프트웨어

Red Hat 솔루션 스택 전체를 뒷받침하는 것은 안전한 소프트웨어 공급망입니다. Red Hat 보안 전략의 초석인 전략적으로 중요한 사례 및 절차의 목표는 보안이 내장되어 있으며 시간이 지남에 따라 지원되는 솔루션을 제공하는 것입니다. Red Hat에서 수행하는 구체적인 단계는 다음과 같습니다.

- 업스트림 관계 및 커뮤니티 참여를 유지하여 처음부터 보안에 집중할 수 있도록 지원합니다.
- 보안 및 성능 트랙 레코드에 따라 패키지 선택 및 구성.
- 관련 소스 코드에서 바이너리를 빌드합니다(단순히 업스트림 빌드를 수락하지 않음).
- 검사 및 품질 보증 도구 세트를 적용하여 광범위한 잠재적 보안 문제 및 회귀 문제를 방지합니다.
- 릴리스된 모든 패키지에 디지털 서명하고 암호화 방식으로 인증된 배포 채널을 통해 배포합니다.
- 패치 및 업데이트 배포를 위한 단일 통합 메커니즘 제공. OpenStack에 기초하여 CC(Common Criteria) 인증을 받은 Red Hat Enterprise Linux 및 KVM 구성 요소. 여기에는 물리적 사이트 방문을 수행하는 타사 감사자 및 인터뷰 직원이 모범 사례(예: 공급망 또는 개발에 대한 준수)가 포함됩니다.

또한 Red Hat은 제품에 대한 위협과 취약점을 분석하고 고객 포털을 통해 관련 조언과 업데이트를 제공하는 전담 보안 팀을 유지하고 있습니다. 이 팀은 대부분의 이론적인 문제와 달리 어떤 문제가 중요한지 결정합니다. Red Hat 제품 보안 팀은 전문 지식을 보유하고 있으며 서브스크립션 제품과 관련된 업스트림 커뮤니티에 기여합니다. 프로세스의 핵심 부분인 Red Hat 보안 공지는 Red Hat 솔루션에 영향을 미치는 보안 결함에 대한 사전 알림을 제공합니다. 또한 취약점이 처음 게시되는 당일에 자주 배포되는 패치와 함께 Red Hat 솔루션에 영향을 미치는 보안 취약점에 대한 사전 알림을 제공합니다.

2장. 문서

설명서에서는 OpenStack 환경에 대한 일반적인 설명을 제공하고 사용되는 모든 시스템(예: 프로덕션, 개발 또는 테스트)을 다루어야 합니다. 시스템 구성 요소, 네트워크, 서비스 및 소프트웨어를 문서화하는 것은 보안 문제, 공격 벡터 및 가능한 보안 영역 브리징 지점을 철저히 다루는 데 필요한 관점을 제공하는 경우가 많습니다. 시스템 인벤토리는 기존 IT 환경에서 영구 리소스일 경우 가상 시스템 또는 가상 디스크 볼륨과 같은 임시 리소스를 캡처해야 할 수 있습니다.

2.1. 시스템 역할 및 유형

사용자 환경에서 사용되는 역할을 문서화합니다. 일반적으로 OpenStack 설치를 구성하는 크게 정의된 두 가지 유형의 노드는 다음과 같습니다.

- **인프라 노드** - OpenStack API 프로바이더(예: neutron), 메시지 대기열 서비스, 스토리지 관리, 모니터링, 네트워킹 및 클라우드의 운영 및 프로비저닝을 지원하는 데 필요한 기타 서비스와 같은 클라우드 관련 서비스를 실행합니다.

- **컴퓨팅, 스토리지 또는 기타 리소스 노드** - 클라우드에서 실행되는 인스턴스의 컴퓨팅 및 스토리지 용량을 제공합니다.

2.2. 하드웨어 인벤토리

서면 문서에 대한 엄격한 규정 준수 요구 사항이 없는 클라우드는 CMDB(구성 관리 데이터베이스)가 있으면 도움이 될 수 있습니다. CMDB는 일반적으로 하드웨어 자산 추적 및 전체 라이프사이클 관리에 사용됩니다. CMDB를 활용함으로써 조직은 계산 노드, 스토리지 노드 또는 네트워크 장치와 같은 클라우드 인프라 하드웨어를 신속하게 식별할 수 있습니다. CMDB는 불완전한 유지 관리, 부적절한 보호로 인해 취약점이 있을 수 있는 네트워크에 존재하는 자산을 식별하는 데 도움이 될 수 있습니다. 기본 하드웨어에서 필수 자동 검색 기능을 지원하는 경우 OpenStack 프로비저닝 시스템은 몇 가지 기본 CMDB 기능을 제공할 수 있습니다.

2.3. 소프트웨어 인벤토리

하드웨어와 마찬가지로 OpenStack 배포 내의 모든 소프트웨어 구성 요소를 문서화해야 합니다. 예를 들면 다음과 같습니다.

- MySQL 또는 mongoDB와 같은 시스템 데이터베이스

- ID 또는 계산과 같은 OpenStack 소프트웨어 구성 요소

- 로드 밸런서, 역방향 프록시, DNS 또는 DHCP 서비스와 같은 지원 구성 요소 지원 구성 요소는 라이브러리, 애플리케이션 또는 소프트웨어 클래스에서 손상 또는 취약점의 영향을 평가할 때 신뢰할 수 있는 소프트웨어 구성 요소 목록입니다.

3장. 암호화 및 키 관리

장치 간 통신은 보안상 중요한 문제입니다. Heartbleed와 같은 중요한 취약점이나 BEAST 및 CRIME 등의 고급 공격으로 입증된 네트워크상의 안전한 커뮤니케이션 방법이 점점 더 중요해지고 있습니다. 그러나 암호화는 더 큰 보안 전략의 일부에 불과합니다. 엔드포인트가 손상되면 공격자가 더 이상 사용된 암호화를 중단할 필요가 없지만 시스템에서 처리할 때 메시지를 보고 조작할 수 있습니다.

이 장에서는 내부 및 외부 리소스를 모두 보호하기 위해 TLS(Transport Layer Security) 구성에 대한 기능을 검토하고 특정 주의가 수행해야 하는 특정 시스템 범주를 호출합니다.

OpenStack 구성 요소는 다양한 프로토콜을 사용하여 서로 통신하며 통신에는 민감한 데이터 또는 기밀 데이터가 포함될 수 있습니다. 공격자는 중요한 정보에 액세스하기 위해 채널을 도청하려고 할 수 있습니다. 따라서 모든 구성 요소는 보안 통신 프로토콜을 사용하여 서로 통신해야 합니다.

3.1. TLS 및 SSL 소개

OpenStack 배포에서 네트워크 트래픽의 기밀성 또는 무결성을 보장하기 위한 보안 요구 사항이 있는 경우가 있습니다. 일반적으로 TLS 프로토콜과 같은 암호화 측정값을 사용하여 이를 구성합니다. 일반적인 배포에서는 공용 네트워크를 통해 전송된 모든 트래픽이 보안 강화되어야 하지만 보안상의 모범 사례에서는 내부 트래픽도 보호해야 합니다. 보호를 위해 보안 영역 분리에 의존하는 것은 충분하지 않습니다. 공격자가 하이퍼바이저 또는 호스트 리소스에 액세스하거나 API 엔드포인트 또는 기타 서비스를 손상시키는 경우 메시지, 명령 또는 기타 관리 기능을 쉽게 삽입하거나 캡처할 수 없어야 합니다.

보안은 관리 영역 서비스 및 서비스 내 통신을 포함하여 TLS로 모든 영역을 강화해야 합니다. TLS는 OpenStack 서비스에 대한 사용자 통신의 인증, 비요청, 기밀성 및 OpenStack 서비스 자체 간의 무결성을 보장하는 메커니즘을 제공합니다.

SSL(Secure Sockets Layer) 프로토콜에서 게시된 취약점으로 인해 더 이상 사용되지 않는 브라우저나 라이브러리와의 호환성이 필요하지 않은 한, SSL보다 TLS 1.2 이상을 사용하는 것이 좋습니다.

3.2. 공개 키 인프라

PKI(Public Key Infrastructure)는 데이터 및 인증 보안을 위한 암호화 알고리즘, 암호 모드 및 프로토콜을 제공하는 프레임워크입니다. 이는 당사자의 ID를 검증하는 동안 트래픽을 암호화할 수 있도록 시스템 및 프로세스 집합으로 구성됩니다. 여기에 설명된 PKI 프로필은 PKIX 실무 그룹에서 개발한 IETF(Internet Engineering Task Force) PKIX(Public Key Infrastructure) 프로파일입니다. PKI의 핵심 구성 요소는 다음과 같습니다.

- 디지털 인증서 - 서명된 공개 키 인증서는 엔터티의 데이터를 확인할 수 있는 데이터 구조이며 다른 일부 속성과 함께 공개 키입니다. 이러한 인증서는 CA(인증 기관)에서 발급합니다. 인증서는 신뢰할 수 있는 CA에서 서명하므로 확인되면 엔터티와 연결된 공개 키가 설명 엔터티와 연결되도록 보장됩니다. 이러한 인증서를 정의하는 데 사용되는 가장 일반적인 표준은 X.509 표준입니다. 현재 표준인 X.509 v3은 RFC5280에 자세히 설명되어 RFC6818에 의해 업데이트됩니다. 인증서는 온라인 엔터티의 ID를 증명하기 위한 메커니즘으로 CA에서 발급됩니다. CA는 인증서에서 메시지 다이제스트를 생성하고 개인 키를 사용하여 다이제스트를 암호화하여 인증서에 디지털 서명합니다.

- End entity - 인증서 대상인 사용자, 프로세스 또는 시스템입니다. 최종 엔터티에서 승인을 위해 인증서 요청을 등록 기관(RA)에 보냅니다. 승인된 경우 RA는 요청을 CA(인증 기관)로 전달합니다. 인증 기관에서 요청을 확인하고 정보가 올바른 경우 인증서가 생성되고 서명됩니다. 이 서명된 인증서는 인증서 리포지토리로 전송됩니다.

- Reliabilitying party - 인증서에 나열된 공개 키에 대한 참조로 확인할 수 있는 디지털 서명 인증서를 수신하는 끝점입니다. 의존하는 당사자는 체인의 인증서를 확인하고 CRL에 없는지 확인하고 인증서의 만료 날짜를 확인할 수 있어야 합니다.

- CA(인증 기관) - CA는 인증 정책, 관리 처리 및 인증서 발급에 대한 인증서에 의존하는 최종 당사자와 당사자가 모두 신뢰할 수 있는 단체입니다.
- 등록 기관(RA) - CA가 특정 관리 기능을 위임하는 선택적 시스템이며, CA에서 인증서를 발급하기 전에 최종 엔터티의 인증과 같은 기능을 포함합니다.
- CRL(인증서 해지 목록) -CRL(인증서 해지 목록)은 해지된 인증서 일련 번호 목록입니다. 이러한 인증서를 제공하는 최종 엔터티는 PKI 모델에서 신뢰해서는 안 됩니다. 호출은 주요 손상, CA 손상과 같은 여러 가지 이유로 발생할 수 있습니다.
- CRL issuer - CA가 인증서 폐기 목록 게시를 위임하는 선택적 시스템입니다.
- 인증서 리포지토리 - 최종 엔터티 인증서 및 인증서 폐기 목록이 저장되고 쿼리되는 위치(인증서 번들이라고도 함)입니다.

3.3. 인증 기관

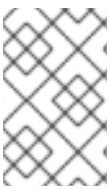
많은 조직에는 자체 CA(인증 기관), 인증 정책 및 내부 OpenStack 사용자 또는 서비스에 대한 인증서를 발급하는 데 사용해야 하는 관리 기능이 있는 기존의 공용 키 인프라가 있습니다. 공용 보안 영역이 인터넷에 연결되어 있는 조직은 널리 인정되는 공용 CA에서 서명한 인증서가 추가로 필요합니다. 관리 네트워크를 통한 암호화 통신의 경우 공용 CA를 사용하지 않는 것이 좋습니다. 대신 대부분의 배포에서 고유한 내부 CA를 배포하는 것이 좋습니다.

참고

TLS를 효과적으로 사용하면 퍼블릭 또는 내부 CA에서 사용할 수 있는 일련의 특정 인증서 문제 또는 와일드카드에서 사용할 수 있는 DNS의 도메인 또는 하위 도메인이 제공되는 배포에 의존합니다. TLS 인증서를 효과적으로 검증할 수 있도록 하려면 플랫폼 서비스에 대한 액세스가 이러한 DNS 레코드를 통해서만 이루어져야 합니다.

OpenStack 클라우드 아키텍트는 내부 시스템 및 고객이 직면하는 서비스를 위해 별도의 PKI 배포를 사용하는 것이 좋습니다. 이를 통해 클라우드 배포자는 PKI 인프라를 지속적으로 제어하고 내부 시스템을 위한 인증서를 요청, 서명 및 배포할 수 있습니다. 고급 구성은 다양한 보안 영역에 별도의 PKI 배포를 사용할 수 있습니다. 이를 통해 OpenStack 운영자는 환경의 암호화 분리를 유지하여 발급된 인증서를 다른 회사에서 인식하지 못하도록 할 수 있습니다.

인터넷을 향한 클라우드 엔드 포인트 (또는 고객이 표준 운영 체제 제공 인증서 번들 이외의 모든 것을 설치할 것으로 예상되는 고객 인터페이스)에서 TLS를 지원하는 데 사용되는 인증서는 운영 체제 인증서 번들에 설치된 인증 기관을 사용하여 프로비저닝해야 합니다.



참고

인증서 생성 및 서명에 관한 관리, 정책 및 기술 문제가 있습니다. 이 영역은 클라우드 설계자나 운영자가 여기에 권장되는 지침 외에도 업계 리더와 벤더의 조언을 찾고자 할 수 있는 영역입니다.

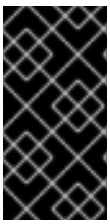
3.4. DIRECTOR를 사용하여 암호화 구성

기본적으로 오버클라우드는 암호화되지 않은 엔드포인트를 서비스에 사용합니다. 즉, 공용 API 엔드포인트에 SSL/TLS를 활성화하려면 오버클라우드 구성에 추가 환경 파일이 필요합니다. *Advanced Overcloud Customization* 가이드에서는 SSL/TLS 인증서를 설정하고 오버클라우드 생성 프로세스의 일부로 포함하는 방법을 설명합니다. https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html/advanced_overcloud_customization/sect-enabling_ssltls_on_the_overcloud

3.5. TLS 라이브러리

OpenStack 에코시스템의 특정 구성 요소, 서비스 및 애플리케이션은 TLS 라이브러리를 사용하도록 구성할 수 있습니다. OpenStack 내의 TLS 및 HTTP 서비스는 일반적으로 FIPS 140-2에 대해 검증된 모듈이 있는 OpenSSL을 사용하여 구현됩니다. 그러나 각 애플리케이션 또는 서비스에서 OpenSSL 라이브러리를 사용하는 방법에 여전히 약점이 될 수 있다고 가정합니다.

3.6. TLS 1.0 사용 중단



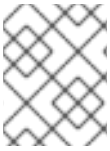
중요

FedRAMP 인증 시스템은 TLS 1.0에서 이동하기 위해 필요합니다. 권장 수준은 1.2이며 광범위한 호환성이 필요한 경우에만 1.1이 허용됩니다. 자세한 내용은 https://www.fedramp.gov/assets/resources/documents/CSP_TLS_Requirements.pdf 의 내용을 참조하십시오.

Red Hat OpenStack Platform 13 배포의 경우 HAProxy에서는 TLS 1.0 연결을 허용하지 않으므로 TLS가 활성화된 API에 대한 TLS 연결을 처리합니다. 이는 **no-tlsv10** 옵션으로 구현됩니다. **InternalTLS** 가 활성화된 HA 배포의 경우 컨트롤러 플레인의 노드 간 트래픽도 암호화됩니다. 여기에는 RabbitMQ, MariaDB 및 Redis가 포함됩니다. MariaDB 및 Redis는 TLS1.0을 더 이상 사용되지 않으며 RabbitMQ에 동일한 사용 중단이 업스트림에서 백포트될 것으로 예상됩니다.

3.6.1. TLS 1.0이 사용 중인지 확인

ciphers **can을 사용하여** 배포에서 TLS 1.0이 제공되는지 여부를 확인할 수 있습니다. https://github.com/mozilla/cipherscan 에서 Cipherscan을 복제할 수 있습니다. 이 예제 출력은 **Horizon** 에서 수신한 결과를 보여줍니다.



참고

처음 실행할 때 추가 종속성을 설치할 수 있으므로 비 프로덕션 시스템에서 ciphers **can** 을 실행합니다.

```
$ ./cipherscan https://openstack.lab.local
...............................
Target: openstack.lab.local:443

prio  ciphersuite                protocols  pfs                curves
1     ECDHE-RSA-AES128-GCM-SHA256  TLSv1.2    ECDH,P-256,256bits  prime256v1
2     ECDHE-RSA-AES256-GCM-SHA384  TLSv1.2    ECDH,P-256,256bits  prime256v1
3     DHE-RSA-AES128-GCM-SHA256    TLSv1.2    DH,1024bits         None
4     DHE-RSA-AES256-GCM-SHA384    TLSv1.2    DH,1024bits         None
5     ECDHE-RSA-AES128-SHA256      TLSv1.2    ECDH,P-256,256bits  prime256v1
6     ECDHE-RSA-AES256-SHA384      TLSv1.2    ECDH,P-256,256bits  prime256v1
7     ECDHE-RSA-AES128-SHA         TLSv1.2    ECDH,P-256,256bits  prime256v1
8     ECDHE-RSA-AES256-SHA         TLSv1.2    ECDH,P-256,256bits  prime256v1
9     DHE-RSA-AES128-SHA256        TLSv1.2    DH,1024bits         None
10    DHE-RSA-AES128-SHA           TLSv1.2    DH,1024bits         None
11    DHE-RSA-AES256-SHA256        TLSv1.2    DH,1024bits         None
12    DHE-RSA-AES256-SHA           TLSv1.2    DH,1024bits         None
13    ECDHE-RSA-DES-CBC3-SHA       TLSv1.2    ECDH,P-256,256bits  prime256v1
14    EDH-RSA-DES-CBC3-SHA         TLSv1.2    DH,1024bits         None
15    AES128-GCM-SHA256            TLSv1.2    None                None
```

```
16    AES256-GCM-SHA384          TLSv1.2    None            None
17    AES128-SHA256              TLSv1.2    None            None
18    AES256-SHA256              TLSv1.2    None            None
19    AES128-SHA                 TLSv1.2    None            None
20    AES256-SHA                 TLSv1.2    None            None
21    DES-CBC3-SHA               TLSv1.2    None            None

Certificate: trusted, 2048 bits, sha256WithRSAEncryption signature
TLS ticket lifetime hint: None
NPN protocols: None
OCSP stapling: not supported
Cipher ordering: server
Curves ordering: server - fallback: no
Server supports secure renegotiation
Server supported compression methods: NONE
TLS Tolerance: yes

Intolerance to:
 SSL 3.254          : absent
 TLS 1.0            : PRESENT
 TLS 1.1            : PRESENT
 TLS 1.2            : absent
 TLS 1.3            : absent
 TLS 1.4            : absent
```

서버를 스캔할 때 Cipherscan은 협상할 의사가 있는 가장 높은 TLS 버전인 특정 TLS 버전에 대한 지원을 알립니다. 대상 서버가 TLS 프로토콜을 올바르게 따르는 경우 상호 지원되는 최고 버전으로 응답합니다. 이 버전은 처음 알린 Cipherscan보다 낮을 수 있습니다. 서버가 특정 버전을 사용하는 클라이언트와의 연결을 계속하는 경우 해당 프로토콜 버전에 적합하지 않은 것으로 간주되지 않습니다. 연결을 설정하지 않으면(지정된 버전 또는 하위 버전 사용) 해당 버전의 프로토콜이 있는 것으로 간주되는 것으로 간주됩니다. 예를 들면 다음과 같습니다.

```
Intolerance to:
 SSL 3.254          : absent
 TLS 1.0            : PRESENT
 TLS 1.1            : PRESENT
 TLS 1.2            : absent
 TLS 1.3            : absent
 TLS 1.4            : absent
```

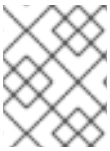
이 출력에서는 **TLS 1.0 및 TLS 1.1** 의 내결함성이 **PRESENT** 로 보고됩니다. 즉, 연결을 설정할 수 없고 해당 TLS 버전에 대한 지원을 광고하는 동안 Cipherscan이 연결할 수 없었습니다. 따라서 스캔한 서버에서 해당(및 하위) 프로토콜 버전이 활성화되지 않았음을 구현하는 것이 타당합니다.

3.7. 암호화 알고리즘, 암호화 모드 및 프로토콜

TLS 1.2만 사용하는 것이 좋습니다. TLS 1.0 및 1.1과 같은 기타 버전은 여러 공격에 취약하며 많은 정부 기관 및 규제 업계에서 명시적으로 금지됩니다. 사용자 환경에서 TLS 1.0을 비활성화해야 합니다. TLS 1.1은 광범위한 클라이언트 호환성에 사용될 수 있지만 이 프로토콜을 활성화할 때 주의해야 합니다. 필수 호환성 요구 사항이 있고 관련 위험을 알고 있는 경우에만 TLS 버전 1.1을 활성화합니다. 모든 버전의 SSL(TLS 이전 버전)은 여러 공용 취약점으로 인해 사용해서는 안 됩니다.

TLS 1.2를 사용하고 클라이언트와 서버를 모두 제어하는 경우 암호화 제품군은 **ECDHE-ECDSA-AES256-GCM-SHA384** 로 제한해야 합니다. 엔드포인트를 모두 제어하지 않고 TLS 1.1 또는 1.2를 사용하는 경우 일반적인 암호는 다음과 같습니다. **!eNULL:!eNULL:!DES:!3DES:!SSLv3:!TLSv1:!CAMELLIA** 는 합리적

인 암호 선택입니다.



참고

이 가이드는 암호화에 대한 참조를 위한 것이 아니며 OpenStack 서비스에서 활성화 또는 비활성화해야 하는 특정 알고리즘 또는 암호 모드에 대한 설명이 아닙니다.

3.8. TLS 프록시 및 HTTP 서비스

OpenStack 엔드포인트는 공용 네트워크 및 관리 네트워크의 다른 OpenStack 서비스에 있는 최종 사용자 모두에 API를 제공하는 HTTP 서비스입니다. 현재 TLS를 사용하여 외부 요청을 암호화할 수 있습니다. Red Hat OpenStack Platform에서 구성하려면 TLS 세션을 설정하고 종료할 수 있는 HAproxy 뒤에 API 서비스를 배포할 수 있습니다.

소프트웨어 종료가 성능이 충분하지 않은 경우 하드웨어 액셀러레이터를 대체 옵션으로 탐색할 필요가 있을 수 있습니다. 이 접근 방식에는 플랫폼의 추가 구성이 필요하며 일부 하드웨어 로드 밸런서가 Red Hat OpenStack Platform과 호환될 수 있는 것은 아닙니다. 선택한 TLS 프록시에서 처리할 요청 크기를 염두에 두는 것이 중요합니다.

3.9. PERFECT FORWARD SECRECY

완벽한 전달 보안을 위해 TLS 서버를 구성하려면 키 크기, 세션 ID 및 세션 티켓을 신중하게 계획해야 합니다. 또한 다중 서버 배포의 경우 공유 상태도 중요한 고려 사항입니다. 실제 배포에서는 성능 향상을 위해 이 기능을 활성화하는 것이 좋습니다. 이 작업은 보안 강화 방식으로 수행할 수 있지만 주요 관리에 대해 특별한 고려가 필요합니다. 이러한 구성은 이 가이드의 범위를 벗어납니다.

3.10. BARBICAN을 사용하여 시크릿 관리

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. Barbican은 현재 다음과 같은 사용 사례를 지원합니다.

- **대칭 암호화 키** - Block Storage(cinder) 볼륨 암호화, 임시 디스크 암호화 및 오브젝트 스토리지(swift) 오브젝트 암호화에 사용됩니다.
- **비대칭 키 및 인증서** - Glance 이미지 서명 및 확인, Octavia TLS 로드 밸런싱.

이번 릴리스에서는 barbican은 cinder, swift, Octavia 및 Compute(nova) 구성 요소와의 통합을 제공합니다. 예를 들어 다음 사용 사례에 barbican을 사용할 수 있습니다.

- **암호화된 볼륨 지원** - barbican을 사용하여 Cinder 암호화 키를 관리할 수 있습니다. 이 구성에서는 LUKS를 사용하여 부팅 디스크를 포함하여 인스턴스에 연결된 디스크를 암호화합니다. 주요 관리 측면은 사용자에게 투명하게 수행됩니다.
- **Glance 이미지 서명** - 업로드한 이미지가 변조되지 않았는지 확인하기 위해 Image 서비스(glance)를 구성할 수 있습니다. 이미지는 barbican에 저장된 키로 먼저 서명되며, 이미지는 매번 사용하기 전에 유효성을 검사합니다.

자세한 내용은 Barbican 가이드 https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html-single/manage_secrets_with_openstack_key_manager/를 참조하십시오.

4장. ID 및 액세스 관리

ID 서비스(keystone)는 클라우드 사용자에게 인증 및 권한 부여를 제공하며 보안 고려 시 중요한 구성 요소입니다.

ID 서비스는 최종 사용자 인증을 직접 제공하거나 외부 인증 방법을 사용하여 조직의 보안 정책 및 요구 사항을 준수하도록 구성할 수 있습니다.

4.1. 인증

인증은 모든 실제 OpenStack 배포에서 필수적인 부분입니다. 시스템 설계의 이러한 측면에 신중하게 생각하십시오. 이 주제를 완전히 처리하는 것은 본 가이드의 범위를 벗어나지만 다음 섹션에 몇 가지 주요 주제가 나와 있습니다.

가장 기본적인 인증은 사용자가 실제로 원하는 ID를 확인하는 프로세스입니다. 시스템에 로그인할 때 사용자 이름과 암호를 제공하는 예입니다.

OpenStack ID 서비스(keystone)는 사용자 이름 및 암호, LDAP 및 기타 외부 인증 방법을 비롯한 여러 인증 방법을 지원합니다. 인증에 성공하면 ID 서비스에서 후속 서비스 요청에 사용되는 권한 부여 토큰을 사용자에게 제공합니다.

TLS(Transport Layer Security)는 X.509 인증서를 사용하는 서비스와 사람 간에 인증을 제공합니다. TLS의 기본 모드는 서버 측 인증이지만, 미국 정부 표준에서 요구되므로 클라이언트 인증에 인증서를 사용해야 합니다.

4.2. 잘못된 로그인 시도

ID 서비스(keystone)는 로그인 시도에 반복적으로 실패한 후 계정으로 액세스를 제한할 수 있습니다. 반복적인 로그인 시도 패턴은 일반적으로 무차별 공격의 지표입니다. 이러한 유형의 공격은 퍼블릭 클라우드 배포에서 더 일반적입니다. 구성된 로그인 시도 횟수가 실패한 후 계정을 차단하는 외부 인증 시스템을 사용하여 이 문제를 완화할 수도 있습니다. 그러면 추가 관리 개입으로만 잠금 해제될 수 있습니다.

탐지 기술을 사용하여 손상을 완화할 수도 있습니다. 감지에는 액세스 제어 로그를 자주 검토하여 계정 액세스 시도를 식별해야 합니다. 가능한 수정에는 사용자 암호의 강점 검토 또는 방화벽 규칙을 통해 공격의 네트워크 소스 차단이 포함됩니다. 연결 수를 제한하는 keystone 서버에 방화벽 규칙을 추가할 수 있습니다. 이렇게 하면 공격의 효과를 줄일 수 있습니다.

또한 비정상적인 로그인 시간 및 의심스러운 작업에 대해 계정 활동을 검사하고 계정 비활성화와 같은 올바른 조치를 취하는 것이 유용합니다.

4.3. 권한 부여

ID 서비스에서는 그룹 및 역할의 개념을 지원합니다. 사용자는 그룹에 속하고 그룹에 역할 목록이 있습니다. OpenStack 서비스는 서비스에 액세스하려는 사용자의 역할을 참조합니다. OpenStack 정책 적용자 미들웨어는 각 리소스와 연관된 정책 규칙을 고려한 다음, 사용자의 그룹/역할 및 연결을 고려하여 요청된 리소스에 액세스가 허용되는지 여부를 결정합니다.

4.4. 기존 액세스 제어 정책 설정

역할, 그룹 및 사용자를 구성하기 전에 OpenStack 설치에 필요한 액세스 제어 정책을 문서화해야 합니다. 정책은 조직의 규정 또는 법적 요구 사항과 일치해야 합니다. 향후 액세스 제어 구성에 대한 수정은 공식 정책에 따라 일관되게 수행되어야 합니다. 정책에는 계정을 생성, 삭제, 비활성화 및 활성화하고 계정에 권한을 할당하기 위한 조건과 프로세스가 포함되어야 합니다. 정책을 정기적으로 검토하고 구성이 승인된 정책을 준수하는지 확인합니다.

4.5. 서비스 권한 부여

클라우드 관리자는 각 서비스에 대해 admin 역할의 사용자를 정의해야 합니다. 이 서비스 계정은 사용자를 인증하기 위한 권한 부여를 서비스에 제공합니다.

ID 서비스를 사용하여 인증 정보를 저장하도록 계산 및 오브젝트 스토리지 서비스를 구성할 수 있습니다. ID 서비스는 활성화될 수 있는 TLS에 대한 클라이언트 인증을 지원합니다. TLS 클라이언트 인증은 사용자 이름 및 암호 외에도 사용자 식별 시 더 나은 안정성을 제공하는 추가 인증 요소를 제공합니다. 사용자 이름과 암호가 손상될 수 있는 경우 무단 액세스의 위험이 줄어듭니다. 그러나 모든 배포에서 불가능할 수 없는 사용자에게 추가 관리 오버헤드와 인증서를 발급하는 비용이 있습니다.

클라우드 관리자는 중요한 구성 파일을 무단 수정으로부터 보호해야 합니다. 이는 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 파일 및 X.509 인증서를 포함하여 SELinux와 같은 필수 액세스 제어 프레임워크를 사용하여 구성할 수 있습니다.

TLS를 사용한 클라이언트 인증에는 서비스에 인증서가 발급되어야 합니다. 이러한 인증서는 외부 또는 내부 인증 기관에서 서명할 수 있습니다. OpenStack 서비스는 기본적으로 신뢰할 수 있는 CA에 대한 인증서 서명의 유효성을 확인하고 서명이 유효하지 않거나 CA가 신뢰할 수 없는 경우 연결이 실패합니다. 클라우드 배포자는 자체 서명 인증서를 사용할 수 있습니다. 이 경우 유효성 검사가 비활성화되거나 인증서가 신뢰할 수 있음으로 표시되어야 합니다. 자체 서명 인증서의 유효성 검사를 비활성화하려면 **/etc/nova/api.paste.ini** 파일의 **[filter:authtoken]** 섹션에 **insecure=False** 를 설정합니다. 이 설정은 다른 구성 요소에 대한 인증서도 비활성화합니다. 컨테이너화된 서비스에는 정확한 파일 경로가 다를 수 있습니다.

4.6. 토큰

사용자가 인증되면 권한을 부여하고 OpenStack 환경에 액세스할 수 있도록 토큰이 생성됩니다. 토큰에는 가변 라이프사이클이 있을 수 있지만 만료 기본값은 1시간입니다. 권장 만료 값은 내부 서비스에서 작업을 완료하는 데 충분한 시간을 허용하는 더 낮은 값으로 설정해야 합니다. 작업이 완료되기 전에 토큰이 만료되는 경우 클라우드가 응답하지 않거나 서비스 제공을 중지할 수 있습니다. 사용 중 사용한 시간의 예로, 계산 서비스에서 로컬 캐싱을 위해 하이퍼바이저에 디스크 이미지를 전송하는 데 필요한 시간이 있습니다.

토큰은 종종 ID 서비스 응답의 큰 컨텍스트 구조 내에서 전달됩니다. 이러한 응답은 또한 다양한 OpenStack 서비스의 카탈로그를 제공합니다. 각 서비스는 이름, 내부, admin 및 공용 액세스의 액세스 엔드포인트로 나열됩니다. ID 서비스는 토큰 취소를 지원합니다. 이 매니페스트는 토큰 취소를 위해 토큰을 취소하고 폐기된 토큰에 대한 쿼리를 캐시하고 캐시된 토큰에서 제거한 다음 캐시된 폐기된 토큰 목록에 동일한 내용을 추가하는 개별 OpenStack 서비스를 나열하기 위한 API로 매니페스트합니다. Fernet 토큰은 Red Hat OpenStack Platform 13에서 지원되는 유일한 토큰입니다.

4.6.1. Fernet 토큰

이제 Fernet 토큰이 기본 토큰 프로바이더입니다. Fernet은 API 토큰에서 사용하도록 명시적으로 설계된 보안 메시징 형식입니다. Fernet 토큰은 비영구적(데이터베이스에 유지되지 않음), 경량(180~ 240바이트)이며 클라우드를 실행하는 데 필요한 운영 오버헤드를 줄입니다. 인증 및 권한 부여 메타데이터를 메시지 페이로드에 번들하여 암호화하여 Fernet 토큰(180~240바이트 내)으로 서명합니다.

UUID, PKI 및 PKIZ 토큰과 달리 Fernet 토큰은 지속성이 필요하지 않습니다. keystone 토큰 데이터베이스는 인증의 부작용으로 인해 더 이상 과도하게 발생하지 않습니다. Fernet 토큰을 사용하는 경우 토큰 데이터베이스에서 만료된 토큰을 정리할 필요가 없습니다. Fernet 토큰은 비영구적이므로 복제할 필요가 없습니다. 각 keystone 노드에서 동일한 리포지토리를 공유하는 경우 노드 간에 Fernet 토큰을 즉시 생성하고 검증할 수 있습니다.

PKI 및 PKIZ 토큰과 비교하여 Fernet 토큰은 크기가 작습니다. 일반적으로 250바이트 제한을 유지합니다. PKI 및 PKIZ 토큰의 경우 더 큰 서비스 카탈로그로 토큰 길이가 길어집니다. 암호화된 페이로드의 콘텐츠가 최소한으로 유지되므로 이 패턴은 Fernet 토큰과 함께 존재하지 않습니다.

Fernet 토큰 및 Fernet 키 순환에 대한 자세한 내용은 https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html-single/deploy_fernet_on_the_overcloud/ 을 참조하십시오.

4.7. KEYSTONE 도메인

Keystone 도메인은 높은 수준의 보안 경계로, 프로젝트, 사용자 및 그룹을 논리적으로 그룹화합니다. 따라서 모든 keystone 기반 ID 구성 요소를 중앙에서 관리하는 데 사용할 수 있습니다. 이제 계정 도메인, 서버, 스토리지 및 기타 리소스를 도입하면서 이제 여러 프로젝트(이전의 테넌트라고 함)로 논리적으로 그룹화할 수 있습니다. 이 프로젝트는 마스터 계정과 유사한 컨테이너 아래에 그룹화할 수 있습니다. 또한 계정 도메인 내에서 여러 사용자를 관리하고 프로젝트마다 다른 역할을 할당할 수 있습니다.

ID V3 API는 여러 도메인을 지원합니다. 다양한 도메인의 사용자는 다양한 인증 백엔드에 표시될 수 있습니다. 다양한 서비스 리소스에 액세스하기 위해 정책 정의에서 사용되는 단일 역할 및 권한 집합에 매핑해야 하는 다양한 특성도 있을 수 있습니다.

규칙이 프로젝트에 속한 admin 사용자와 사용자에게만 액세스를 지정할 수 있는 경우 매핑은 단순할 수 있습니다. 다른 시나리오에서는 클라우드 관리자가 프로젝트당 매핑 루틴을 승인해야 할 수 있습니다.

도메인별 인증 드라이버를 사용하면 도메인별 구성 파일을 사용하여 여러 도메인에 대한 ID 서비스를 구성할 수 있습니다. 드라이버를 활성화하고 도메인별 구성 파일 위치를 설정하는 것은 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 파일의 **[ID]** 섹션에서 발생합니다. 예를 들면 다음과 같습니다.

```
[identity]
domain_specific_drivers_enabled = True
domain_config_dir = /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/
```

도메인별 구성 파일이 없는 모든 도메인은 기본 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 파일의 옵션을 사용합니다.

4.8. LDAP를 사용한 인증

외부 ID 공급자(IdP)를 사용하여 OpenStack 서비스 공급자(SP)에 인증할 수 있습니다. 이 경우 서비스 프로바이더는 OpenStack 클라우드에서 제공하는 서비스입니다.

인증 및 권한 부여 서비스의 경우 OpenStack ID 모델은 외부 인증 데이터베이스를 별도의 keystone 도메인으로 간주합니다. 각 외부 인증 메커니즘은 여러 개의 공존 도메인을 지원하는 keystone 도메인과 연결됩니다. 역할을 사용하여 외부 도메인의 사용자에게 클라우드의 리소스에 대한 액세스 권한을 부여할 수 있습니다. 이 방법은 도메인 간 다중 프로젝트 배포에서도 작동합니다. 또한 모든 OpenStack 역할을 외부에서 인증된 사용자에게 매핑할 수 없으므로 이 방법은 구성별 정책에도 영향을 미칩니다. 예를 들어 외부 인증 데이터베이스의 사용자가 admin **도메인** 의 **admin** 사용자와 유사한 관리 액세스 권한이 필요한 경우 일반적으로 추가 구성 및 고려 사항이 필요합니다.

외부 인증에서는 추가 ID를 프로비저닝하거나 여러 번 로그인할 필요 없이 기존 자격 증명을 사용하여 여러 인증 정보 집합을 사용하여 여러 엔드포인트에서 서버, 볼륨 및 데이터베이스와 같은 클라우드 리소스에 액세스할 수 있습니다. 자격 증명은 사용자의 ID 프로바이더가 유지 관리합니다.

ID 서비스는 사용자 자격 증명을 SQL 데이터베이스에 저장하거나 LDAP 호환 디렉터리 서버를 사용할 수 있습니다. ID 데이터베이스는 저장된 자격 증명의 손상 위험을 줄이기 위해 다른 OpenStack 서비스에서 사용하는 데이터베이스와 분리될 수 있습니다.

사용자 이름과 암호를 사용하여 인증할 때 ID는 암호 강도, 만료 또는 실패한 인증 시도에 정책을 적용하지 않습니다. 더 강력한 암호 정책을 적용하려는 조직은 ID 확장 또는 외부 인증 서비스를 사용하는 것이 좋습니다.

LDAP는 조직의 기존 디렉터리 서비스 및 사용자 계정 관리 프로세스에 ID 인증 통합을 간소화합니다. OpenStack의 인증 및 권한 부여 정책은 다른 서비스에 위임될 수 있습니다. 일반적인 사용 사례는 사설 클라우드를 배포하려고 하며 LDAP 시스템에 직원 및 사용자의 데이터베이스가 이미 있는 조직입니다. 이를 인증 기관으로 사용하면 ID 서비스에 대한 요청이 LDAP 시스템에 위임되므로 정책에 따라 인증 또는 거부됩니다. 인증에 성공하면 ID 서비스에서 권한 있는 서비스에 액세스하는 데 사용되는 토큰을 생성합니다.

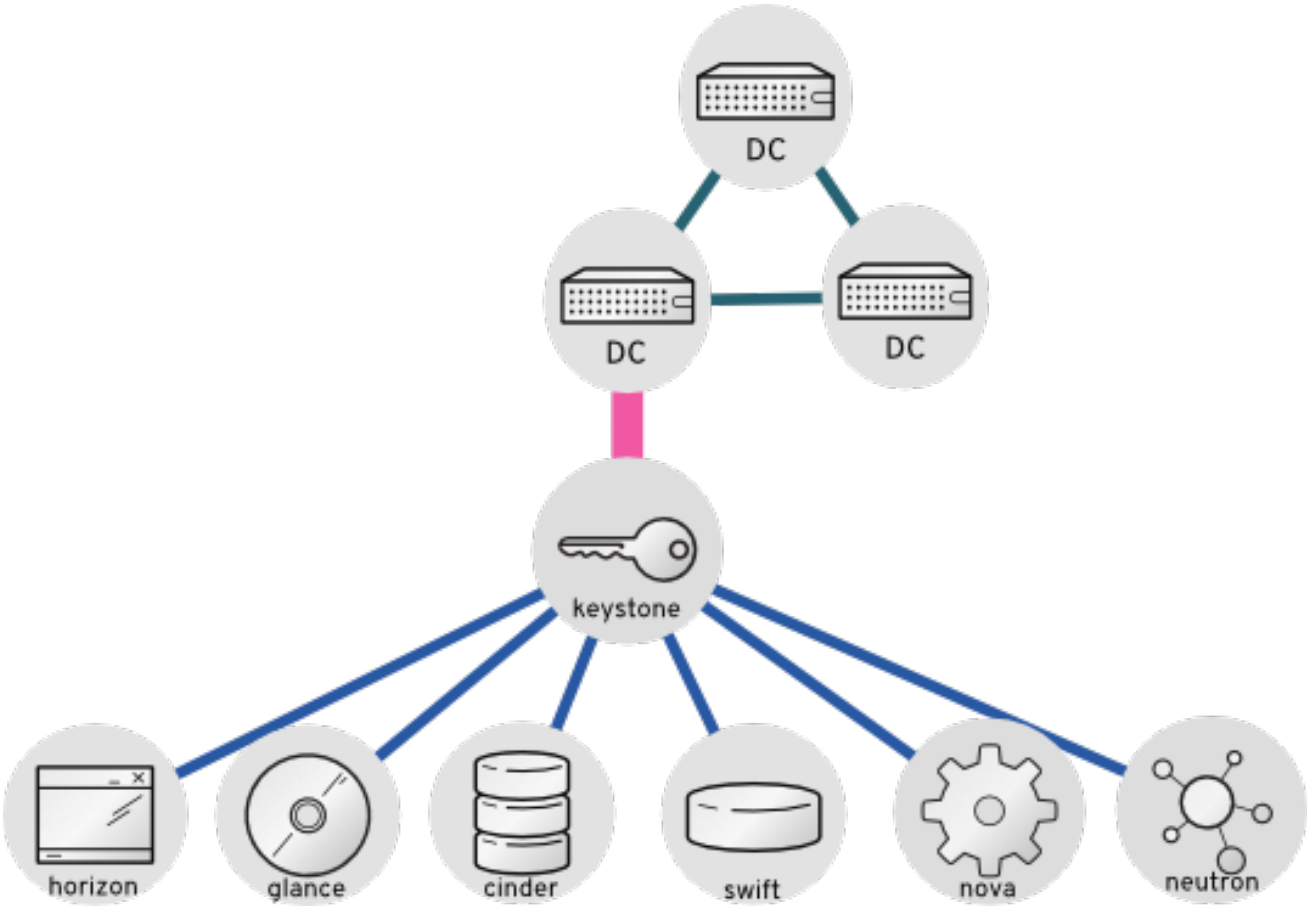
LDAP 시스템에 admin, finance, HR 등과 같은 사용자에 대해 정의된 속성이 있는 경우 다양한 OpenStack 서비스에서 사용할 수 있도록 ID 내의 역할 및 그룹에 매핑해야 합니다. **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 파일은 LDAP 속성을 ID 속성에 매핑합니다.

4.9. LDAP 기반 서비스와 통합

ID 서비스(keystone)는 LDAP 기반 서비스에 저장된 사용자 계정(예: Microsoft AD DS(Active Directory Domain Services), Red Hat IdM(Identity Management))을 인증할 수 있습니다. 이 사용 사례에서 keystone은 LDAP 사용자 데이터베이스 인증에 대한 읽기 전용 액세스 권한을 가지며 인증된 계정에 할당된 authZ 권한에 대한 관리를 유지합니다. authZ 기능(권한, 역할, 프로젝트)은 여전히 keystone에서 수행합니다. 여기서 keystone 관리 툴을 사용하여 LDAP 계정에 권한 및 역할이 할당됩니다.

4.9.1. LDAP 통합 작동 방식

아래 다이어그램에서 keystone은 암호화된 LDAPS 연결을 사용하여 Active Directory 도메인 컨트롤러에 연결합니다. 사용자가 Horizon에 로그인하면 keystone에서 제공된 사용자 자격 증명을 수신하여 authZ의 Active Directory에 전달합니다.



OpenStack을 AD DS 및 IdM과 통합하는 방법에 대한 자세한 내용은 통합 가이드 https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html-single/integrate_with_identity_service/를 참조하십시오.

5장. POLICIES

각 OpenStack 서비스에는 액세스 정책에 의해 관리되는 리소스가 포함되어 있습니다. 예를 들어 리소스에 다음 기능이 포함될 수 있습니다.

- 인스턴스를 생성하고 시작할 수 있는 권한
- 인스턴스에 볼륨을 연결하는 기능

RHOSP(Red Hat OpenStack Platform) 관리자는 다양한 액세스 수준의 새 역할을 도입하거나 기존 역할의 기본 동작을 변경하려면 사용자 지정 정책을 생성해야 할 수 있습니다. 또한 API 액세스 정책을 변경한 후 검증하고 작동하지 않는 경우 해당 정책을 디버깅하는 기능이 필요합니다. 구문 오류가 다운타임으로 이어질 수 있고 잘못된 권한 부여로 인해 보안 또는 유용성에 부정적인 영향을 줄 수 있는 프로덕션 배포 외부에서 정책을 검증 및 디버깅합니다.

5.1. 기존 정책 검토

서비스에 대한 정책 파일은 기존에 **/etc/$service** 디렉토리에 있습니다. 예를 들어 Compute(nova)에 대한 **policy.json** 파일의 전체 경로는 **/etc/nova/policy.json** 이었습니다.

기존 정책을 찾는 방법에 영향을 주는 두 가지 중요한 아키텍처 변경 사항이 있습니다.

- Red Hat OpenStack Platform이 이제 컨테이너화되었습니다.
  - 서비스 컨테이너 내부에서 정책 파일을 보는 경우 정책 파일이 기존 경로에 있습니다. /etc/$service/policy.json
  - 서비스 컨테이너 외부에서 정책 파일을 보는 경우 정책 파일이 다음과 같은 경로에 있습니다. /var/lib/config-data/puppet-generated/$service/etc/$service/policy.json
- 각 서비스에는 코드로 제공된 기본 정책, 수동으로 생성한 경우에만 사용할 수 있는 파일 또는 with **oslopolicy** 툴링이 생성되는 경우에만 사용할 수 있습니다. 정책 파일을 생성하려면 다음 예와 같이 컨테이너 내에서 the **oslopolicy-policy-generator** 를 사용합니다.

```
docker exec -it keystone oslopolicy-policy-generator --namespace keystone
```

기본적으로 생성된 정책은 osly.policy CLI 툴을 통해 stdout로 푸시됩니다.

5.2. 서비스 정책 이해

서비스 정책 파일 문은 별칭 정의 또는 규칙입니다. 별칭 정의는 파일의 맨 위에 있습니다. 다음 목록에는 Compute(nova)용 **policy.json** 파일의 별칭 정의에 대한 설명이 포함되어 있습니다.

- "context_is_admin": "role:admin"
  타겟 뒤에 **rule:context_is_admin** 이 나타나면 해당 작업을 허용하기 전에 사용자가 관리 컨텍스트에서 작동하는지 확인합니다.
- "admin_or_owner": "is_admin:True or project_id:%(project_id)s"
  대상 뒤에 **admin_or_owner** 가 표시되면 정책에서 사용자가 admin인지 또는 해당 작업을 허용하기 전에 해당 오브젝트의 소유 프로젝트 ID와 일치하는지 확인합니다.
- "admin_api": "is_admin:True
  타겟 뒤에 **admin_api** 가 나타나면 정책에서 해당 작업을 허용하기 전에 사용자가 admin인지 확인합니다.

5.3. 정책 구문

Policy.json 파일은 이러한 설정의 대상 범위를 제어할 수 있도록 특정 연산자를 지원합니다. 예를 들어 다음 keystone 설정에는 admin 사용자만 사용자를 만들 수 있는 규칙이 포함되어 있습니다.

```
"identity:create_user": "rule:admin_required"
```

: 문자 왼쪽에 있는 섹션은 권한을 설명하고 오른쪽에 있는 섹션에서는 권한을 사용할 수 있는 사람을 정의합니다. Operator를 오른쪽에 사용하여 범위를 추가로 제어할 수도 있습니다.

- **!** - 사용자(관리자 포함)가 이 작업을 수행할 수 없습니다.
- **@** 및 **""** - 모든 사용자가 이 작업을 수행할 수 있습니다.
- **및 또는 - 표준 연산자 기능을 사용할 수 있습니다.**

예를 들어 다음 설정은 사용자가 새 사용자를 생성할 수 있는 권한이 없음을 의미합니다.

```
"identity:create_user": "!"
```

5.4. 액세스 제어에 정책 파일 사용

기본 규칙을 재정의하려면 적절한 OpenStack 서비스에 대한 policy.json 파일을 편집합니다. 예를 들어 계산 서비스에는 nova 디렉터리에 policy.json 이 있으며, 컨테이너 내부에서 볼 때 컨테이너화된 서비스에 대한 파일의 올바른 위치입니다.

참고

- **프로덕션에 구현하기 전에 스테이징 환경에서 정책 파일에 대한 변경 사항을 철저하게 테스트해야 합니다.**
- **액세스 제어 정책에 대한 변경 사항이 의도적으로 리소스의 보안을 약화하지 않는지 확인해야 합니다. 또한 policy.json 파일을 변경하는 것은 즉시 유효하며 서비스를 다시 시작할 필요가 없습니다.**

5.5. 예제: 파워 유저 역할 생성

keystone 역할의 권한을 사용자 지정하려면 서비스의 policy.json 파일을 업데이트합니다. 즉, 사용자 클래스에 할당한 권한을 더 세분화하여 정의할 수 있습니다. 이 예제에서는 다음 권한을 사용하여 배포에 대한 *power 사용자* 역할을 생성합니다.

- 인스턴스를 시작합니다.
- 인스턴스를 중지합니다.
- 인스턴스에 연결된 볼륨을 관리합니다.

이 역할의 목적은 관리자 액세스 권한을 부여할 필요 없이 특정 사용자에게 추가 권한을 부여하는 것입니다. 이러한 권한을 사용하려면 사용자 지정 역할에 다음 권한을 부여해야 합니다.

- 인스턴스 시작: "os_compute_api:servers:start": "role:PowerUsers"
- 인스턴스를 중지합니다. "os_compute_api:servers:stop": "role:PowerUsers"
- 특정 볼륨을 사용하도록 인스턴스 구성: "os_compute_api:servers:create:attach_volume": "role:PowerUsers"
- 인스턴스에 연결된 볼륨을 나열합니다. "os_compute_api:os-volumes-attachments:index": "role:PowerUsers"
- 볼륨 연결: "os_compute_api:os-volumes-attachments:create": "role:PowerUsers"
- 연결된 볼륨의 세부 정보 보기: "os_compute_api:os-volumes-attachments:show": "role:PowerUsers"
- 인스턴스에 연결된 볼륨을 변경합니다. "os_compute_api:os-volumes-attachments:update": "role:PowerUsers"
- 인스턴스에 연결된 볼륨을 삭제합니다. "os_compute_api:os-volumes-attachments:delete": "role:PowerUsers"

참고

policy.json 파일을 수정할 때 기본 정책을 재정의합니다. 결과적으로 **PowerUsers** 의 멤버는 이러한 작업을 수행할 수 있는 유일한 사용자입니다. **admin** 사용자가 이러한 권한을 유지할 수 있도록 하려면 ***admin_or_power_user에 대한 규칙을 만들 수 있습니다.*** 또한 몇 가지 기본적인 조건 논리를 사용하여 역할을 정의할 수 있습니다.**PowerUsers** 또는 **role:Admin.**

1.

명령줄 세션에서 **keystone v3 API**를 사용하려면 **v3** 끝점 및 설정을 정의하는 **rc** 파일을 가져옵니다.

```
OS_AUTH_URL=http://controller-hostname.lab.local:5000/v3
OS_USERNAME=username
OS_PASSWORD=password
OS_USER_DOMAIN_NAME=Default
OS_PROJECT_DOMAIN_NAME=Default
OS_PROJECT_NAME=project-name
OS_IDENTITY_API_VERSION=3
```

2.

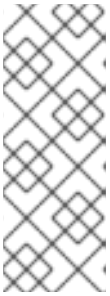
사용자 지정 **keystone** 역할을 만듭니다.

```
$ openstack role create PowerUsers
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 7061a395af43455e9057ab631ad49449 |
| name      | PowerUsers                       |
+-----------+----------------------------------+
```

3.

기존 사용자를 역할에 추가하고 역할을 프로젝트에 할당합니다.

```
$ openstack role add --project [PROJECT_NAME] --user [USER_ID] [PowerUsers-ROLE_ID]
```



참고

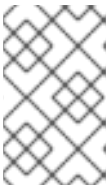
역할 할당은 하나의 프로젝트와 독점적으로 쌍으로 구분됩니다. 즉, 사용자에게 역할을 할당하는 경우 대상 프로젝트를 동시에 정의합니다. 사용자에게 동일한 역할을 받으려면 다른 프로젝트에 대해 역할을 다시 할당해야 하지만 다른 프로젝트를 대상으로 해야 합니다.

4.

기본 **nova** 정책 설정을 확인합니다.

```
$ oslopolicy-policy-generator --namespace nova
```

5. **/var/lib/config-data/puppet-generated/nova/etc/nova/policy.json**에 다음 항목을 추가하여 새 **PowerUsers** 역할에 대한 사용자 지정 권한을 생성합니다.



참고

배포 전에 정책 변경을 테스트하여 예상대로 작동하는지 확인하십시오.

```
{
"os_compute_api:servers:start": "role:PowerUsers",
"os_compute_api:servers:stop": "role:PowerUsers",
"os_compute_api:servers:create:attach_volume": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:index": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:create": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:show": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:update": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:delete": "role:PowerUsers"
}
```

이 파일을 저장할 때 변경 사항을 구현하고 **nova** 컨테이너를 다시 시작합니다. **PowerUsers keystone** 역할에 추가된 사용자는 이러한 권한을 받습니다.

5.6. 예제: 속성에 따라 액세스 제한

해당 **API** 호출을 수행하는 사용자의 특성에 따라 **API** 호출에 대한 액세스를 제한하는 정책을 생성할 수 있습니다. 예를 들어 다음 기본 규칙은 키 쌍 삭제가 관리 컨텍스트에서 실행되는 경우 또는 토큰의 사용자 **ID**가 대상과 연결된 사용자 **ID**와 일치하는 경우를 나타냅니다.

```
"os_compute_api:os-keypairs:delete": "rule:admin_api or user_id:%(user_id)s"
```

참고: * 새로 구현된 기능은 각 릴리스의 모든 서비스에서 보장되지 않습니다. 따라서 타겟 서비스의 기존 정책에 대한 규칙을 사용하여 규칙을 작성하는 것이 중요합니다. 이러한 정책 보기에 대한 자세한 내용은 기존 정책 검토를 참조하십시오. * 모든 정책은 여러 릴리스 버전과의 호환성에 대해 정책이 보장되지 않으므로 배포되는 모든 버전에 대해 비 프로덕션 환경에서 엄격하게 테스트해야 합니다.

위 예제를 기반으로 리소스를 소유하는지 여부에 따라 사용자에 대한 액세스를 확장하거나 제한하도록 **API** 규칙을 생성할 수 있습니다. 또한 속성을 다른 제한 사항과 결합하여 아래 예와 같이 규칙을 형성할 수 있습니다.

```
"admin_or_owner": "is_admin:True or project_id:%(project_id)s"
```

위의 예제를 고려하면 관리자와 사용자로 제한된 고유한 규칙을 생성한 다음 해당 규칙을 사용하여 작업을 추가로 제한할 수 있습니다.

```
"admin_or_user": "is_admin:True or user_id:%(user_id)s"
"os_compute_api:os-instance-actions": "rule:admin_or_user"
```

사용 가능한 **policy.json** 구문 옵션에 대한 자세한 내용은 정책 구문을 참조하십시오.

5.7. 사용자 및 역할 감사

Red Hat OpenStack Platform에서 사용할 수 있는 툴을 사용하여 사용자별 역할 할당 보고서와 관련 권한 보고서를 작성할 수 있습니다.

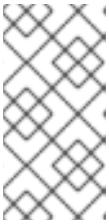
1. **openstack role list** 명령을 실행하여 현재 사용자 환경에서 역할을 확인합니다.

```
openstack role list -c Name -f value

swiftoperator
ResellerAdmin
admin
_member_
heat_stack_user
```

2. **openstack role assignment list** 명령을 실행하여 특정 역할의 멤버인 모든 사용자를 나열합니다. 예를 들어 **admin** 역할이 있는 모든 사용자를 보려면 다음을 실행합니다.

```
$ openstack role assignment list --names --role admin
+-------+-------------------------------------+-------+-----------------+------------+--------+-----------+
| Role  | User                                | Group | Project         | Domain     | System | Inherited |
+-------+-------------------------------------+-------+-----------------+------------+--------+-----------+
| admin | heat-cfn@Default                    |       | service@Default |            |        | False     |
| admin | placement@Default                   |       | service@Default |            |        | False     |
| admin | neutron@Default                     |       | service@Default |            |        | False     |
| admin | zaqar@Default                       |       | service@Default |            |        | False     |
| admin | swift@Default                       |       | service@Default |            |        | False     |
| admin | admin@Default                       |       | admin@Default   |            |        | False     |
| admin | zaqar-websocket@Default             |       | service@Default |            |        | False     |
| admin | heat@Default                        |       | service@Default |            |        | False     |
| admin | ironic-inspector@Default            |       | service@Default |            |        | False     |
| admin | nova@Default                        |       | service@Default |            |        | False     |
| admin | ironic@Default                      |       | service@Default |            |        | False     |
| admin | glance@Default                      |       | service@Default |            |        | False     |
| admin | mistral@Default                     |       | service@Default |            |        | False     |
| admin | heat_stack_domain_admin@heat_stack  |       |                 | heat_stack |        | False
```

```
|
| admin | admin@Default                    |      |                |           | all    | False    |
+-------+----------------------------------+------+----------------+-----------+--------+----------+
```



참고

f {csv,json,table,value,yaml} 매개변수를 사용하여 이러한 결과를 내보낼 수 있습니다.

5.8. API 액세스 감사

지정된 역할이 액세스할 수 있는 **API** 호출을 감사할 수 있습니다. 각 역할에 대해 이 프로세스를 반복하면 각 역할에 대해 액세스 가능한 **API**에 대한 포괄적인 보고서가 생성됩니다. 다음 단계의 경우 다음을 수행합니다.

- 타겟 역할에서 사용자로 가져올 인증 파일입니다.
- **JSON** 형식의 액세스 토큰입니다.
- 감사하려는 각 서비스의 **API**에 대한 정책 파일입니다.

절차

1. 원하는 역할에서 사용자의 인증 파일을 가져와 시작합니다.
2. **Keystone** 생성된 토큰을 캡처하여 파일에 저장합니다. **openstack-cli** 명령을 실행하고 제공된 토큰을 **stdout**에 출력하는 **--debug** 옵션을 사용하여 이 작업을 수행할 수 있습니다. 이 토큰을 복사하여 액세스 파일에 저장할 수 있습니다. 다음 명령을 사용하여 단일 단계로 이 작업을 수행합니다.

```
openstack token issue --debug 2>&1 | egrep ^'{\"token\":' > access.file.json
```

3. 정책 파일을 만듭니다. 컨테이너화된 서비스를 호스팅하는 오버클라우드 노드에서 이 작업을 수행할 수 있습니다. 다음 예제에서는 **cinder** 서비스에 대한 정책 파일을 생성합니다.

```
ssh heat-admin@CONTROLLER-1 sudo docker exec cinder_api \
oslopolicy-policy-generator \
--config-file /etc/cinder/cinder.conf \
```

```
--namespace cinder > cinderpolicy.json
```

4. **이제 이러한 파일을 사용하여 cinder API에 액세스하기 위해 해당 파일에서 역할을 감사할 수 있습니다.**

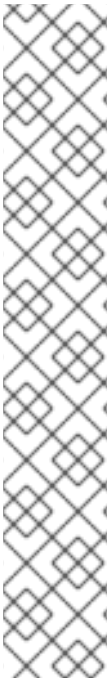
```
oslopolicy-checker --policy cinderpolicy.json --access access.file.json
```

6장. 인프라 및 가상화 강화

운영 절차에는 새로운 취약점과 보안 업데이트에 대해 배울 계획이 있어야 합니다. 하드웨어 및 소프트웨어 벤더는 일반적으로 취약점의 존재를 알립니다. 이러한 문제를 해결하기 위한 대안과 패치를 제공할 수 있습니다.

Red Hat Product Security는 보안 업데이트를 계속 인식할 수 있도록 사이트를 유지 관리합니다.

- https://www.redhat.com/mailman/listinfo/rhsa-announce
- https://www.redhat.com/wapps/ugc/protected/notif.html
- https://access.redhat.com/security/security-updates/#/?q=openstack&p=1&sort=portal_publication_date%20desc&rows=10&portal_advisory_type=Security%20Advisory&documentKind=PortalProduct



참고

업데이트 추적 외에도 프로세스 및 배포에서 일반 보안 업데이트 설치를 수용할 수 있는지 확인합니다. 다음 지침도 사용하십시오.

- 프로세스를 설계할 때 인스턴스 보안 업데이트를 포함합니다.
- 커널 업데이트를 위해 컴퓨팅 및 관리 노드를 재부팅합니다.
- 새로 생성된 인스턴스에 최신 업데이트가 있도록 호스팅 이미지 서비스(**glance**) 이미지를 정기적으로 업데이트합니다.

6.1. 하이퍼바이저

하이퍼바이저 플랫폼을 평가할 때 하이퍼바이저가 실행될 하드웨어의 지원 가능성을 고려하십시오. 또한 하드웨어에서 사용할 수 있는 추가 기능과 **OpenStack** 배포의 일부로 선택한 하이퍼바이저에서 이러한 기능을 어떻게 지원하는지 살펴보겠습니다. 이를 위해 하이퍼바이저에는 각각 고유한 **HCL**(하드웨어 호환성 목록)이 있습니다. 호환 하드웨어를 선택할 때는 보안 측면에서 하드웨어 기반 가상화 기술이 중요한지 미리 알아야 합니다.

6.1.1. 하이퍼바이저와 베어 메탈 비교

KVM과 같은 하이퍼바이저를 사용하는 것과 비교하여 Linux 컨테이너 또는 베어 메탈 시스템 사용의 차이점을 인식하는 것이 중요합니다. 특히, 이 보안 가이드의 초점은 하이퍼바이저와 가상화 플랫폼을 기반으로 합니다. 그러나 구현에 베어 메탈 또는 컨테이너화된 환경을 사용해야 하는 경우 해당 환경 배포와 관련하여 특정 차이점에 주의해야 합니다.

베어 메탈의 경우 다시 프로비저닝 및 해제하기 전에 데이터가 올바르게 삭제되었는지 확인합니다. 또한 노드를 재사용하기 전에 하드웨어가 변조되지 않았거나 손상되지 않았음을 보장해야 합니다. 자세한 내용은 https://docs.openstack.org/ironic/queens/admin/cleaning.html에서 참조하십시오.

6.1.2. 하이퍼바이저 메모리 최적화

특정 하이퍼바이저는 메모리 최적화 기술을 사용하여 게스트 가상 머신에 메모리를 과다 할당할 수 있습니다. 매우 복잡한 컴퓨팅 클러스터를 배포할 수 있는 유용한 기능입니다. 이 기술에 대한 한 가지 접근 방식은 메모리 페이지 중복 제거 또는 공유를 통해 이루어집니다. 메모리에 두 가상 시스템에 동일한 데이터가 있는 경우 동일한 메모리를 참조하도록 하는 이점이 있습니다. 일반적으로 이 작업은 커널 동일 페이지 병합(KSM)과 같은 COW(Copy-On-Write) 메커니즘을 통해 수행됩니다. 이러한 메커니즘은 공격에 취약합니다.

- 메모리 중복 제거 시스템은 사이드 채널 공격에 취약합니다. 교육 연구에서 공격자는 옆의 가상 시스템에서 실행되는 소프트웨어 패키지 및 버전을 식별할 수 있었으며, 공격자 VM에서 메모리 액세스 시간을 분석하여 소프트웨어 다운로드 및 기타 중요한 정보를 확인할 수 있었습니다. 결과적으로 한 VM은 모든 프로젝트를 신뢰하거나 동일한 수준의 신뢰 수준을 공유하는 멀티 프로젝트 환경에 적합하지 않을 수 있는 다른 VM의 상태를 추측할 수 있습니다.

- 보다 중요한 점은, 실행 가능한 메모리의 교차 VM 수정을 시행하기 위해 KSM에 대한 행 해머 유형 공격이 입증되었습니다. 즉, hostile 인스턴스가 동일한 Compute 호스트의 다른 인스턴스에 대한 코드 실행 액세스 권한을 얻을 수 있습니다.

배포자는 강력한 프로젝트 분리가 필요한 경우 (공용 클라우드 및 일부 프라이빗 클라우드) KSM을 비활성화해야 합니다.

- KSM을 비활성화하려면 KSM 비활성화를 참조하십시오.

6.2. PCI 통과

PCI 패스스루를 사용하면 인스턴스에서 노드의 하드웨어 조각에 직접 액세스할 수 있습니다. 예를 들어, 이 방법을 사용하여 고성능 계산을 위해 컴퓨팅 CUDA(통합 장치 아키텍처)를 제공하는 비디오 카드

또는 GPU에 액세스할 수 있습니다. 이 기능은 직접 메모리 액세스 및 하드웨어 제약의 두 가지 유형의 보안 위험을 따릅니다.

DMA(직접 메모리 액세스)는 특정 하드웨어 장치가 호스트 컴퓨터의 임의 물리적 메모리 주소에 액세스할 수 있는 기능입니다. 종종 비디오 카드에는 이 기능이 있습니다. 그러나 동일한 노드에서 실행되는 호스트 시스템과 기타 인스턴스를 모두 볼 수 있으므로 인스턴스에 임의의 실제 메모리 액세스 권한이 부여되지 않아야 합니다. 하드웨어 벤더는 IOMMU(입력/출력 메모리 관리 단위)를 사용하여 이러한 상황에서 DMA 액세스를 관리합니다. 하이퍼바이저가 이 하드웨어 기능을 사용하도록 구성되어 있는지 확인해야 합니다.

하드웨어는 인스턴스가 펌웨어 또는 장치의 다른 일부 부분을 악의적으로 수정할 때 발생합니다. 이 장치는 다른 인스턴스 또는 호스트 OS에서 사용하므로 악의적인 코드가 해당 시스템으로 확산될 수 있습니다. 결과적으로 하나의 인스턴스에서 보안 영역 외부에서 코드를 실행할 수 있습니다. 이는 가상 하드웨어보다 물리적 하드웨어의 상태를 재설정하기 어렵기 때문에 보안 침해가 발생하고 관리 네트워크에 대한 액세스와 같은 추가 노출로 이어질 수 있습니다.

PCI 통과와 관련된 위험 및 복잡성 때문에 기본적으로 비활성화해야 합니다. 특정 요구 사항에 맞게 활성화된 경우 재사용하기 전에 하드웨어가 깨끗한지 확인하기 위해 적절한 프로세스를 마련해야 합니다.

6.3. SELINUX

필수 액세스는 QEMU 프로세스의 권한을 필요한 항목으로만 제한하여 시도한 공격에 미치는 영향을 제어합니다. Red Hat OpenStack Platform에서 SELinux는 각 QEMU 프로세스를 별도의 보안 컨텍스트에서 실행하도록 구성되어 있습니다. Red Hat OpenStack Platform 서비스를 위해 SELinux 정책이 미리 구성되어 있습니다.

OpenStack의 SELinux 정책은 하이퍼바이저 호스트와 가상 시스템을 두 가지 주요 위협 요인으로부터 보호하는 데 도움이 됩니다.

- 하이퍼바이저 위협 - 가상 시스템 내에서 실행되는 손상된 애플리케이션이 하이퍼바이저를 공격하여 기본 리소스에 액세스합니다. 예를 들어 가상 시스템이 하이퍼바이저 OS, 물리적 장치 또는 기타 애플리케이션에 액세스할 수 있는 경우. 이 위협 벡터는 하이퍼바이저가 물리적 하드웨어를 손상시키고 다른 가상 시스템 및 네트워크 세그먼트를 노출할 수 있기 때문에 상당한 위험을 나타냅니다.

- 가상 시스템(다중 프로젝트) 위협 - VM 내에서 실행되는 손상된 애플리케이션이 하이퍼바이저를 공격하여 다른 가상 시스템과 해당 리소스에 액세스하거나 제어합니다. 이는 가상화에 고유한 위협적 벡터이며 단일 애플리케이션의 취약점으로 인해 여러 개의 가상 시스템 파일 이미지가 손상될 수 있기 때문에 상당한 위험을 나타냅니다. 실제 네트워크를 보호하기 위한 관리 기술이 가상 환경에 직접 적용되지 않으므로 이 가상 네트워크 공격은 주요 우려 사항입니다. 각 KVM 기반 가상 시스템은 SELinux에서 레이블이 지정된 프로세스로, 각 가상 시스템 주변의 보안 경계를

효과적으로 설정합니다. 이 보안 경계는 Linux 커널에서 모니터링 및 시행하므로 호스트 시스템 데이터 파일 또는 기타 VM과 같은 경계 외부 리소스에 대한 가상 시스템의 액세스를 제한합니다.

Red Hat의 SELinux 기반 격리는 가상 시스템 내부에서 실행되는 게스트 운영 체제와 관계없이 제공됩니다. Linux 또는 Windows VM을 사용할 수 있습니다.

6.3.1. 레이블 및 카테고리

KVM 기반 가상 머신 인스턴스는 svirt_image_t 라는 자체 SELinux 데이터 유형으로 레이블이 지정됩니다. 커널 수준 보호는 웨어웨어와 같은 무단 시스템 프로세스가 디스크에서 가상 시스템 이미지 파일을 조작하지 못하도록 방지합니다. 가상 머신의 전원이 꺼지면 이미지는 다음과 같이 svirt_image_t 로 저장됩니다.

```
system_u:object_r:svirt_image_t:SystemLow image1
system_u:object_r:svirt_image_t:SystemLow image2
system_u:object_r:svirt_image_t:SystemLow image3
system_u:object_r:svirt_image_t:SystemLow image4
```

svirt_image_t 레이블 은 디스크에서 이미지 파일을 고유하게 식별하여 SELinux 정책에서 액세스를 제한할 수 있습니다. KVM 기반 컴퓨팅 이미지의 전원이 켜지면 SELinux는 이미지에 임의 숫자 식별자를 추가합니다. SELinux는 하이퍼바이저 노드당 최대 524,288개의 가상 시스템에 숫자 식별자를 할당할 수 있지만 대부분의 OpenStack 배포에서는 이러한 제한사항이 발생하지 않습니다. 이 예에서는 SELinux 범주 식별자를 보여줍니다.

```
system_u:object_r:svirt_image_t:s0:c87,c520 image1
system_u:object_r:svirt_image_t:s0:419,c172 image2
```

6.3.2. SELinux 사용자 및 역할

SELinux는 사용자 역할을 관리합니다. 이러한 항목은 -Z 플래그를 통해 확인하거나 semanage 명령을 사용하여 볼 수 있습니다. 하이퍼바이저에서 관리자만 시스템에 액세스할 수 있어야 하며 관리 사용자와 시스템에 있는 기타 사용자 모두에 적절한 컨텍스트가 있어야 합니다.

6.4. 컨테이너화된 서비스

nova, glance, keystone과 같은 특정 서비스가 이제 컨테이너 내에서 실행됩니다. 이 접근 방식은 서비스에 업데이트를 더 쉽게 적용하여 보안 포스처를 개선하는 데 도움이 됩니다. 자체 컨테이너에서 각 서비스를 실행하면 동일한 베어 메탈에서 공존하는 서비스 간의 격리도 향상됩니다. 이는 연결된 서비스에 쉽게 액세스하지 못하여 한 서비스가 공격에 취약해야 공격 면적을 줄이는 데 도움이 될 수 있습니다.

참고

컨테이너에 마운트된 호스트 시스템의 경로는 **ro**/**rw**로 구성된 경우 컨테이너와 호스트 간에 데이터를 전송하는 데 마운트 지점으로 사용할 수 있습니다.

컨테이너화된 서비스가 임시인 경우 구성 파일을 업데이트하려는 경우 고려해야 하는 특정 관리 방법이 있습니다.

- 물리적 노드의 호스트 운영 체제(예: **/etc/cinder/cinder.conf**)에서 찾을 수 있는 구성 파일을 업데이트하지 마십시오. 컨테이너화된 서비스에서 이 파일을 참조하지 않기 때문입니다.
- 컨테이너 내에서 실행 중인 구성 파일을 업데이트하지 마십시오. 컨테이너를 재시작하면 변경 사항이 손실되기 때문입니다.

대신 컨테이너화된 서비스에 변경 사항을 추가해야 하는 경우 컨테이너를 시드하는 데 사용되는 구성 파일을 업데이트해야 합니다. 이러한 파일은 초기 배포 중 **puppet**을 통해 생성되며, 클라우드 실행에 중요한 중요한 데이터를 포함하므로 적절하게 처리해야 합니다. 이러한 파일은 **/var/lib/config-data/puppet-generated/** 에 저장됩니다. 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 시작하면 이러한 파일에 대한 변경 사항이 적용됩니다.

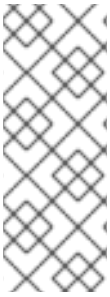
6.5. 컴퓨팅 배포 강화

OpenStack 배포와 관련된 주요 보안 문제 중 하나는 **/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf** 파일과 같은 중요한 파일에 대한 보안 및 제어입니다. 이 구성 파일에는 구성 세부 정보 및 서비스 암호를 포함하여 중요한 많은 옵션이 포함되어 있습니다. 이러한 중요한 파일에는 엄격한 파일 수준 권한이 부여되어야 하며 **AIDE**와 같은 파일 무결성 모니터링(**FIM**) 도구를 통해 변경 사항이 있는지 모니터링해야 합니다. 이러한 유틸리티는 알려진 양호한 상태의 대상 파일을 해시한 다음 주기적으로 파일의 새 해시를 사용하여 알려진 양호한 해시와 비교합니다. 가 예기치 않게 수정된 것으로 확인된 경우 경고를 생성할 수 있습니다.

파일이 포함된 디렉토리로 이동하여 **ls -lh** 명령을 실행하여 파일의 권한을 검사할 수 있습니다. 그러면 파일에 대한 액세스 권한이 있는 권한, 소유자 및 그룹과 파일이 수정된 마지막 시간 및 생성된 시기와 같은 기타 정보가 표시됩니다.

/var/lib/nova 디렉터리에는 지정된 컴퓨팅 노드의 인스턴스에 대한 정보가 들어 있습니다. 이 디렉터리는 엄격하게 적용되는 파일 권한이 있는 중요한 것으로 간주되어야 합니다. 또한 해당 호스트와 연결된 인스턴스의 정보 및 메타데이터가 포함되어 있으므로 정기적으로 백업해야 합니다.

배포에 전체 가상 머신 백업이 필요하지 않은 경우 해당 노드에서 실행되는 각 인스턴스의 결합된 공간만큼 **/var/lib/nova/instances** 디렉터리를 제외하는 것이 좋습니다. 배포에 전체 **VM** 백업이 필요한 경우 이 디렉터리가 성공적으로 백업되었는지 확인해야 합니다.



참고

블록 스토리지(**cinder**) 볼륨에 사용되는 스토리지 하위 시스템(예: **Ceph**)에 저장된 데이터는 네트워크 또는 논리적 액세스를 허용하는 경우 스토리지 하위 시스템에서 전체 가상 시스템 이미지를 검색하여 잠재적으로 **OpenStack** 제어를 우회할 수 있기 때문에 중요한 것으로 간주해야 합니다.

6.6. 펌웨어 업데이트

물리적 서버는 복잡한 펌웨어를 사용하여 서버 하드웨어 및 경량 관리 카드를 활성화하고 작동하므로 자체 보안 취약점이 있을 수 있어 시스템 액세스 및 중단이 발생할 수 있습니다. 이를 해결하기 위해 하드웨어 벤더는 운영 체제 업데이트와 별도로 설치된 펌웨어 업데이트를 발행합니다. 펌웨어 업데이트에 물리적 호스트를 재부팅해야 하는 경우가 있음을 확인하여 이러한 업데이트를 정기적으로 검색, 테스트 및 구현하는 운영 보안 프로세스가 필요합니다.

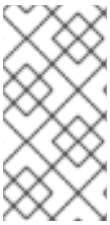
6.7. 블록 스토리지

OpenStack 블록 스토리지(**cinder**)는 소프트웨어(서비스 및 라이브러리)를 제공하여 영구적인 블록 수준 스토리지 장치를 셀프 서비스로 관리하는 서비스입니다. 이렇게 하면 **Compute(nova)** 인스턴스와 함께 사용할 블록 스토리지 리소스에 대한 온디맨드 액세스가 생성됩니다. 이를 통해 블록 스토리지 풀을 소프트웨어 구현 또는 기존 하드웨어 스토리지 제품 중 하나일 수 있는 다양한 백엔드 스토리지 장치로 가상화하여 소프트웨어 정의 스토리지를 생성합니다. 이 기능의 주요 기능은 블록 장치의 생성, 첨부 및 분리를 관리하는 것입니다. 소비자는 백엔드 스토리지 장비 유형 또는 해당 장비가 있는 위치에 대한 정보가 필요하지 않습니다.

컴퓨팅 인스턴스는 **iSCSI, ATA over Ethernet** 또는 **Fibre-Channel**과 같은 업계 표준 스토리지 프로토콜을 사용하여 블록 스토리지를 저장하고 검색합니다. 이러한 리소스는 **OpenStack** 네이티브 표준 **HTTP RESTful API**를 사용하여 관리 및 구성됩니다.

6.7.1. 볼륨 와이핑

블록 스토리지 장치를 초기화하는 방법에는 여러 가지가 있습니다. 기존 접근 방식은 **lvm_type** 을 **thin**로 설정한 다음 **volume_ clear** 매개 변수를 사용하는 것입니다. 또는 볼륨 암호화 기능이 **us ed**이면 볼륨 암호화 키가 삭제되면 볼륨 삭제가 필요하지 않습니다.



참고

이전에는 **lvm_type=default** 를 사용하여 초기화를 나타냅니다. 이 방법은 여전히 작동하지만 보안 삭제를 설정하는 데 **lvm_type=default** 는 권장되지 않습니다.

volume_ clear 매개 변수는 인수로 **0** 또는 **shred** 를 허용할 수 있습니다. **0** 은 하나의 장치에 **0**을 전달합니다. 분할된 작업은 미리 결정된 비트 패턴의 세 가지 전달을 작성합니다.

6.7.2. config 파일의 사용자/그룹 소유권을 root/cinder로 설정

구성 파일에는 구성 요소의 원활한 작동에 필요한 중요한 매개 변수 및 정보가 포함되어 있습니다. 의도하거나 실수로 권한이 없는 사용자가 매개 변수 또는 파일 자체를 수정하거나 삭제하면 심각한 가용성 문제로 인해 다른 최종 사용자에게 서비스가 거부됩니다. 따라서 이러한 중요한 구성 파일의 사용자 소유권을 **root**로 설정하고 그룹 소유권을 **cinder** 로 설정해야 합니다.

cinder 그룹은 호스트에 없습니다. ***ls -l*** 을 사용하면 그룹 소유자의 **GID**가 표시됩니다.

```
sudo ls -l /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf
-rw-r-----. 1 root 42407 188012 Dec 11 09:34 /var/lib/config-data/puppet-
generated/cinder/etc/cinder/cinder.conf

sudo stat -L -c "%U %G" /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf
root UNKNOWN
```

다음 명령으로 **cinder.conf** 구성 파일의 사용자 및 그룹 소유권이 각각 **root** 및 **cinder**로 설정되어 있는지 확인합니다.

```
sudo docker exec -it cinder_api stat -L -c "%U %G" /etc/cinder/cinder.conf
root cinder
```

6.7.3. 구성 파일에 대한 엄격한 권한 설정

cinder.conf 구성 파일에 대한 권한이 **640** 또는 **stricter**로 설정되어 있는지 확인합니다.

```
$ stat -L -c "%a" /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf
```

6.7.4. 인증에 keystone 사용

/var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf 에서 [DEFAULT] 섹션 아래의 auth_strategy 값이 keystone 으로 설정되어 있고 noauth 가 아닌 auth_strategy 값이 설정되어 있는지 확인합니다.

6.7.5. 인증을 위해 TLS 활성화

/var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf 에서 [keystone _authtoken] 섹션 아래의 auth_ uri 값이 https:// '로 시작하는 ID API 엔드포인트로 설정되어 있으며 [keystone_authtoken] 아래의 'insecure] 도 False 로 설정되어 있는지 확인합니다.

6.7.6. 블록 스토리지가 TLS를 사용하여 컴퓨팅과 통신하는지 확인

cinder.conf 에서 [DEFAULT] 섹션 아래의 glance_api_servers 값이 https:// 로 시작하는 값으로 설정되어 있는지 확인하고, glance_api_insecure 매개 변수의 값이 False 로 설정되어 있는지 확인합니다.

6.7.7. 요청 본문에 대한 최대 크기 설정

요청당 최대 본문 크기가 정의되지 않은 경우 공격자는 대규모의 임의의 OSAPI 요청을 작성할 수 있으므로 서비스가 충돌하고 결국 서비스 거부 공격이 발생합니다. t를 최대값으로 할당하면 악의적인 과다 크기 요청으로 인해 서비스의 지속적인 가용성이 차단됩니다.

cinder.conf 의 [oslo _middleware] 섹션 아래의 max_ request_body_size 가 114688 로 설정되어 있는지 확인합니다.

6.7.8. 볼륨 암호화 활성화

암호화되지 않은 볼륨 데이터를 사용하면 공격자가 다양한 VM의 데이터를 읽을 수 있으므로 볼륨 호스팅 플랫폼이 공격자의 가치를 높입니다. 또한 물리적 저장 매체 cou ld는 다른 시스템에서 도난, 다시 마운트 및 액세스합니다. 볼륨 데이터 및 볼륨 백업을 암호화하면 이러한 위험을 완화하고 볼륨 호스팅 플랫폼에 대한 심층적인 방어를 제공할 수 있습니다. 블록 스토리지(c 인더)는 디스크에 기록되기 전에 볼륨 데이터를 암호화할 수 있으므로 볼륨 암호화를 활성화하고 개인 키 스토리지에 Barbican을 사용하는 것이 좋습니다.

6.8. 네트워킹

OpenStack Networking 서비스(neutron)를 사용하면 최종 사용자 또는 프로젝트에서 네트워킹 리소스를 정의하고 사용할 수 있습니다. OpenStack Networking은 클라우드의 인스턴스에 대한 네트워크 연결 및 IP 주소 지정을 정의하는 프로젝트 방향 API를 제공하는 동시에 네트워크 구성을 오케스트레이션합니다. API 중심 네트워킹 서비스로의 전환을 통해 클라우드 설계자와 관리자는 물리 및 가상 네트워크 인프라 및 서비스를 보호하기 위한 모범 사례를 고려해야 합니다.

OpenStack Networking은 오픈 소스 커뮤니티 또는 타사 서비스를 통해 API의 확장성을 제공하는 플러그인 아키텍처를 사용하여 설계되었습니다. 아키텍처 설계 요구 사항을 평가할 때 OpenStack Networking 핵심 서비스에서 사용할 수 있는 기능, 타사 제품이 제공하는 추가 서비스 및 물리적 인프라에 구현하기 위해 필요한 추가 서비스를 결정하는 것이 중요합니다.

이 섹션에서는 OpenStack 네트워킹을 구현할 때 고려해야 하는 프로세스 및 모범 사례에 대한 간략한 개요입니다.

6.8.1. 네트워킹 아키텍처

OpenStack Networking은 여러 노드에 여러 프로세스를 배포하는 독립 실행형 서비스입니다. 이러한 프로세스는 서로 다른 OpenStack 서비스와 상호 작용합니다. OpenStack Networking 서비스의 주요 프로세스는 OpenStack Networking API를 노출하고 추가 처리를 위해 프로젝트 요청을 플러그인 제품군에 전달하는 Python 데몬인 neutron-server입니다.

OpenStack Networking 구성 요소는 다음과 같습니다.

- neutron 서버(neutron-server 및 neutron-*-plugin) - neutron-server 서비스는 컨트롤러 노드에서 실행되고 Networking API 및 해당 확장 기능(또는 플러그인)을 서비스합니다. 또한 각 포트의 네트워크 모델 및 IP 주소 지정을 적용합니다. neutron-server는 영구 데이터베이스에 직접 액세스할 수 있어야 합니다. 에이전트는 AMQP(Advanced Message Queuing Protocol)를 사용하여 통신하는 neutron-server를 통해 데이터베이스에 간접적으로 액세스할 수 있습니다.

- Neutron 데이터베이스 - 데이터베이스는 Neutron 정보의 중앙 집중식 소스이며, 데이터베이스는 데이터베이스의 모든 트랜잭션을 기록하는 API입니다. 이렇게 하면 여러 Neutron 서버가 동일한 데이터베이스 클러스터를 공유하여 모두 동기화되고 네트워크 구성 토폴로지의 지속성을 유지할 수 있습니다.

- 플러그 인 에이전트(neutron-*-agent) - 각 계산 노드 및 네트워킹 노드(L3 및 DHCP 에이전트와 함께)에서 실행하여 로컬 가상 스위치(vswitch) 구성을 관리합니다. 활성화된 플러그인은 활성화되는 에이전트를 결정합니다. 이러한 서비스에는 메시지 큐 액세스 및 사용 중인 플러그인에 따라 외부 네트워크 컨트롤러 또는 SDN 구현에 대한 액세스가 필요합니다. OpenDaylight(ODL) 및 OVN(Open Virtual Network)과 같은 일부 플러그인에서는 계산 노드에 python 에이전트가 필요하지 않으므로 통합을 위해 Neutron 플러그인만 활성화해야 합니다.

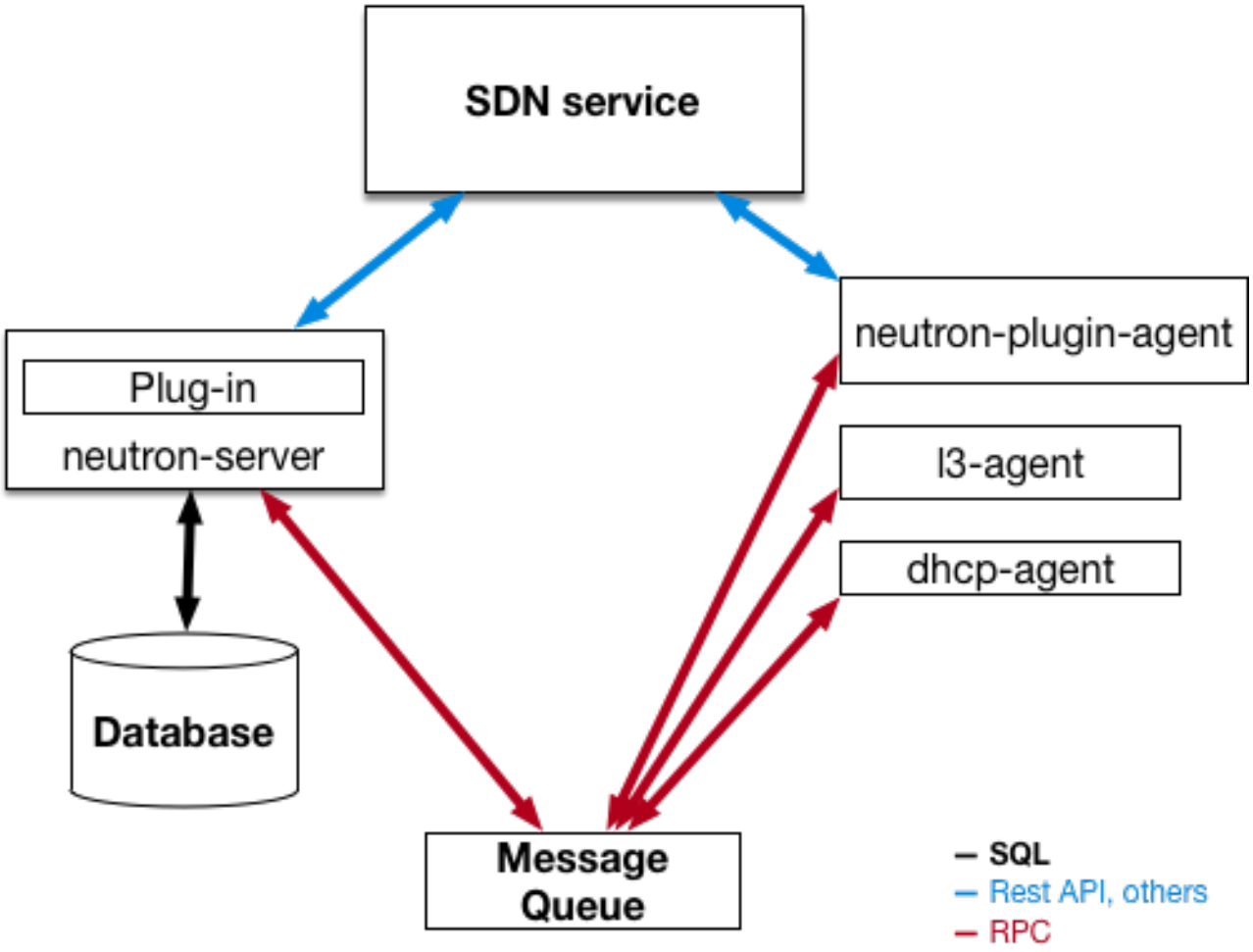
- **DHCP 에이전트(neutron-dhcp-agent) -** 프로젝트 네트워크에 **DHCP** 서비스를 제공합니다. 이 에이전트는 모든 플러그인에서 동일하며 **DHCP** 구성을 유지 관리합니다. **neutron-dhcp-agent**에는 메시지 큐 액세스 권한이 필요합니다. 플러그인에 따라 선택 사항입니다.

- **metadata agent(neutron-metadata-agent,neutron-ns-metadata-proxy) -** 인스턴스 운영 체제 구성 및 사용자 제공 초기화 스크립트(**'userdata'**)를 적용하는 데 사용되는 메타데이터 서비스를 제공합니다. 구현을 수행하려면 **cloud -init**에서 전송한 메타데이터 **API** 요청을 메타데이터 에이전트에 프록시하기 위해 **L3** 또는 **DHCP** 에이전트 네임스페이스에서 **neutron-ns-metadata-proxy** 를 실행해야 합니다.

- **L3 에이전트(neutron-l3-agent) -** 프로젝트 네트워크에서 **VM**의 외부 네트워크 액세스를 위해 **L3/NAT** 전달을 제공합니다. 메시지 큐 액세스가 필요합니다. 플러그인에 따라 선택 사항입니다.

- 네트워크 프로바이더 서비스(**SDN** 서버/서비스) **-** 프로젝트 네트워크에 추가 네트워킹 서비스를 제공합니다. 이러한 **SDN** 서비스는 **REST API**와 같은 통신 채널을 통해 **neutron-server, neutron-plugin** 및 **plugin-agents**와 상호 작용할 수 있습니다.

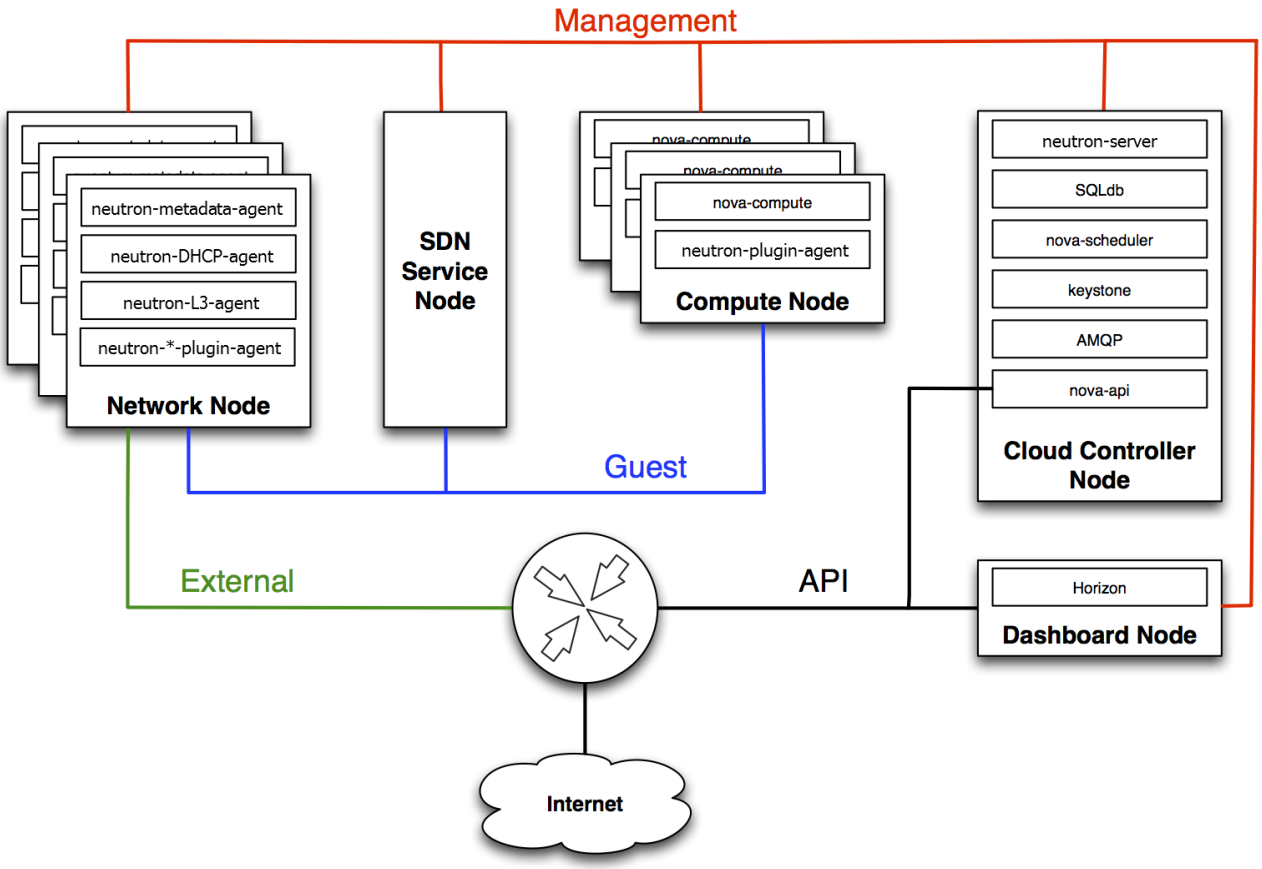
다음 다이어그램에서는 **OpenStack Networking** 구성 요소의 아키텍처 및 네트워킹 흐름 다이어그램을 보여줍니다.



DVR(Distributed Virtual Routing) 및 **L3HA(High Availability)**를 사용하면 이 접근 방식이 크게 변경됩니다. **L3HA**는 라우터 간에 **VRRP**를 구현하므로 이러한 모드는 **neutron**의 보안 환경을 변경합니다. **VRRP** 스푸핑의 위협을 해결하려면 배포의 크기를 올바르게 조정하고 강화하여 라우터 간 로컬 네트워크 트래픽을 민감하게 처리해야 합니다. **DVR**은 네트워크 노드가 필요한 동안 네트워킹 구성 요소(예: 라우팅)를 컴퓨팅 노드로 이동합니다. 따라서 컴퓨팅 노드에는 공용 네트워크에 대한 액세스 권한이 있어야 하므로, 방화벽 규칙 및 보안 모델이 이 접근 방식을 지원하는지 확인해야 하므로 노출을 늘리고 추가적인 보안 고려가 필요합니다.

6.8.2. 물리적 서버에서 Neutron 서비스 배치

이 섹션에서는 컨트롤러 노드, 네트워크 노드 및 인스턴스를 실행하는 계산 노드 집합을 포함하는 표준 아키텍처에 대해 설명합니다. 물리적 서버에 대한 네트워크 연결을 설정하기 위해 일반적인 **neutron** 배포에는 최대 **4**개의 개별 물리적 데이터 센터 네트워크가 있습니다.



- 관리 네트워크 - OpenStack 구성 요소 간 내부 통신에 사용됩니다. 이 네트워크의 IP 주소는 데이터 센터 내에서만 연결할 수 있어야 하며 Management Security(관리 보안) 영역으로 간주됩니다. 기본적으로 관리 네트워크 역할은 *내부 API 네트워크에서* 수행합니다.

- 게스트 네트워크 - 클라우드 배포 내에서 VM 데이터 통신에 사용됩니다. 이 네트워크의 IP 주소 지정 요구 사항은 사용 중인 OpenStack Networking 플러그인과 프로젝트에서 만든 가상 네트워크의 네트워크 구성 선택 사항에 따라 다릅니다. 이 네트워크는 게스트 보안 영역으로 간주됩니다.

- 외부 네트워크 - 일부 배포 시나리오에서 VM에 대한 인터넷 액세스를 제공하는 데 사용됩니다. 이 네트워크의 IP 주소는 인터넷의 모든 사용자가 연결할 수 있어야 합니다. 이 네트워크는 공용 보안 영역에 있는 것으로 간주됩니다. 이 네트워크는 neutron *External* 네트워크에서 제공합니다. 이러한 neutron VLAN은 외부 브리지에서 호스팅됩니다. Red Hat OpenStack Platform director는 Red Hat OpenStack Platform director가 생성하지 않지만 배포 후 neutron에 의해 생성됩니다.

- 공용 API 네트워크 - OpenStack Networking API를 포함한 모든 OpenStack API를 프로젝트에 노출합니다. 이 네트워크의 IP 주소는 인터넷의 모든 사용자가 연결할 수 있어야 합니다. 이 네트워크는 외부 네트워크와 동일한 네트워크일 수 있습니다. IP 할당 범위를 사용하는 외부 네트워크의 서브넷을 IP 블록의 전체 범위보다 작은 서브넷을 만들 수 있기 때문입니다. 이 네트워크는 공용 보안 영역에 있는 것으로 간주됩니다.

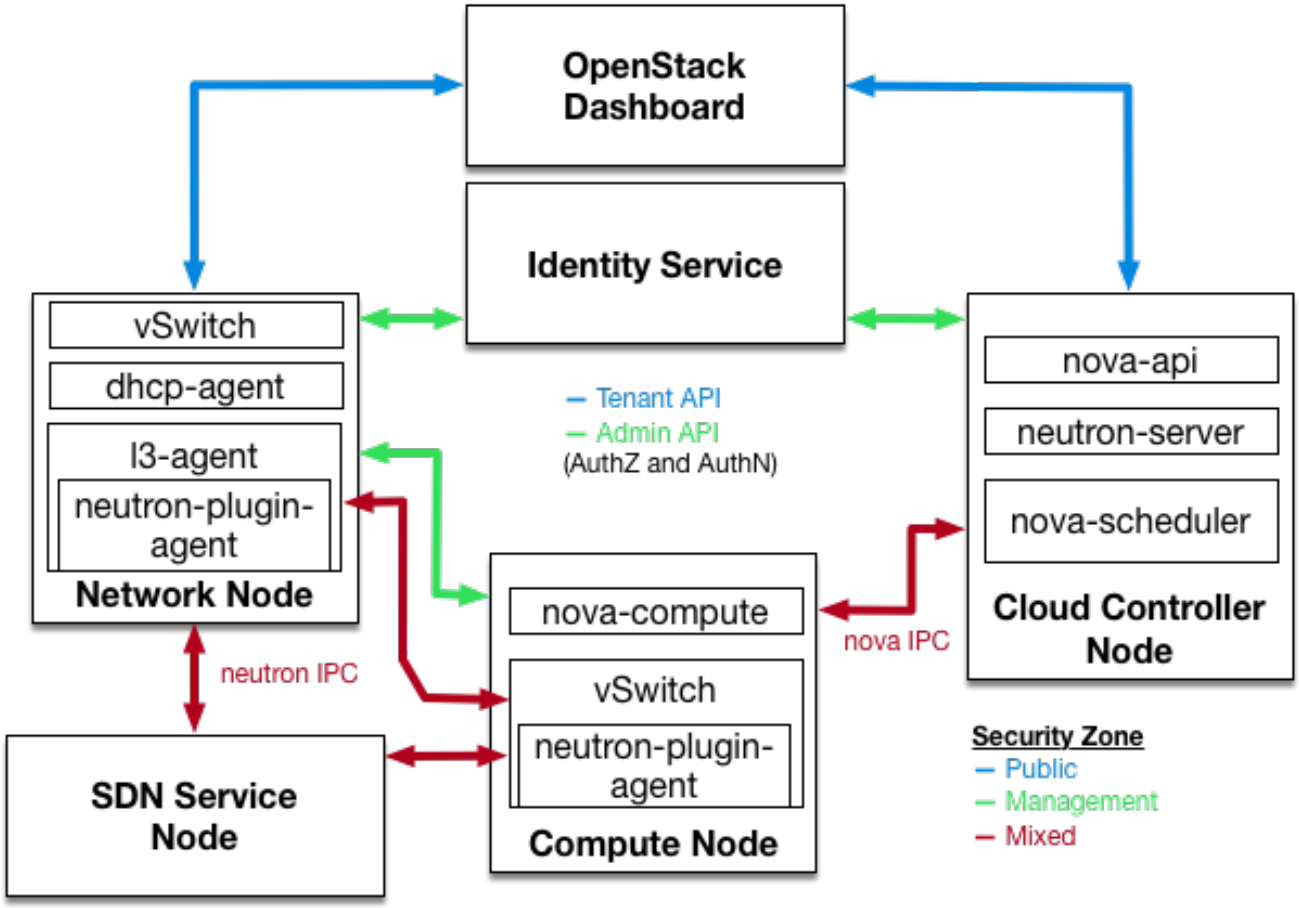
이 트래픽을 별도의 영역으로 분할하는 것이 좋습니다. 자세한 내용은 다음 섹션을 참조하십시오.

6.8.3. 보안 영역

중요한 시스템을 서로 분리하기 위해 보안 영역 개념을 사용하는 것이 좋습니다. 실제로는 **VLAN** 및 방화벽 규칙을 사용하여 네트워크 트래픽을 격리하는 것을 의미합니다. 이 **sho uld**는 세부적인 세부 정보로 수행되며, 결과적으로 **neutron**에 연결해야 하는 서비스만 이를 수행할 수 있습니다.

다음 다이어그램에서는 특정 구성 요소를 분리하기 위해 영역이 생성되었음을 확인할 수 있습니다.

- 대시보드: 공용 네트워크 및 관리 네트워크에 액세스할 수 있습니다.
- **Keystone:** 관리 네트워크에 액세스 가능.
- 계산 노드: 관리 네트워크 및 계산 인스턴스에 액세스할 수 있습니다.
- 네트워크 노드: 사용 중인 **neutron-plugin**에 따라 관리 네트워크, 계산 인스턴스 및 공용 네트워크에 액세스할 수 있습니다.
- **SDN** 서비스 노드: 사용된 제품 및 구성에 따라 관리 서비스, 계산 인스턴스 및 공용이 제공될 수 있습니다.



.

6.8.4. 네트워킹 서비스

OpenStack 네트워크 인프라를 설계하는 초기 아키텍처 단계에서 적절한 보안 제어 및 감사 메커니즘을 식별하기 위해 **ee** 물리적 네트워킹 인프라를 설계하는 데 도움이 되는 적절한 전문 지식을 확보하는 것이 중요합니다.

OpenStack Networking은 프로젝트에 고유한 가상 네트워크를 설계할 수 있는 기능을 제공하는 가상화된 네트워크 서비스 계층을 추가합니다. 현재 이러한 가상화된 서비스는 기존 네트워킹보다 성숙하지 않습니다. 가상화된 서비스의 현재 상태를 고려하십시오. 가상화된 서비스의 현재 상태가 가상화되고 기존 네트워크 경계에서 구현해야 하는 컨트롤을 지정하기 때문에 이를 채택하기 전에.

6.8.5. VLAN 및 터널링을 사용하는 L2 격리

OpenStack Networking은 프로젝트/네트워크 조합에서 트래픽 분리를 위해 서로 다른 두 가지 메커니즘을 사용할 수 있습니다. **VXLAN** 또는 **GRE** 캡슐화를 사용하는 **VLAN(IEEE 802.1Q** 태그 지정) 또는 **L2** 터널. **OpenStack** 배포의 범위와 규모는 트래픽 분리 또는 격리에 사용해야 하는 방법을 결정합니다.

6.8.6. VLAN

VLAN은 특정 VLAN ID(VID) 필드 값이 있는 IEEE 802.1Q 헤더를 포함하는 특정 물리적 네트워크에서 패킷으로 인식됩니다. 동일한 실제 netw ork를 공유하는 VLAN 네트워크는 L2에서 서로 격리되며 겹치는 IP 주소 공간을 가질 수도 있습니다. VLAN 네트워크를 지원하는 개별 실제 네트워크는 고유한 VID 값이 있는 별도의 VLAN trun k로 처리됩니다. 유효한 VID 값은 1 ~ 4094입니다.

VLAN 구성 복잡성은 OpenStack 설계 요구 사항에 따라 다릅니다. OpenStack 네트워킹이 VLAN을 더 효율적으로 사용할 수 있도록 하려면 VLAN 범위(각 프로젝트당 하나씩)를 할당하고 각 컴퓨팅 노드 실제 스위치 포트를 VLAN 트렁크 포트로 설정해야 합니다.

6.8.7. L2 터널링

네트워크 터널링은 해당 조합에 속하는 네트워크 트래픽을 식별하는 데 사용되는 고유한 "tunnel-id"와 각 프로젝트/네트워크 조합을 캡슐화합니다. 프로젝트의 L2 네트워크 연결은 물리적 지역 또는 기본 네트워크 설계와 독립적입니다. IP 패킷 내부에 트래픽을 캡슐화하면 해당 트래픽이 계층-3 경계를 통과하여 사전 구성된 VLAN 및 VLAN 트렁킹이 필요하지 않습니다. 터널링은 네트워크 데이터 트래픽에 난독화 계층을 추가하여 개별 프로젝트 트래픽의 가시성을 줄이고 모니터링 지점을 끊습니다.

OpenStack Networking은 현재 GRE 및 VXLAN 캡슐화를 모두 지원합니다. L2 격리를 제공하는 기술 선택은 배포에 생성될 프로젝트 네트워크의 범위와 크기에 따라 달라집니다.

6.8.8. 액세스 제어 목록

계산에서는 OpenStack Networking 서비스를 사용하여 프로젝트 네트워크 트래픽 액세스 제어를 지원합니다. 보안 그룹을 사용하면 관리자와 프로젝트에서 가상 인터페이스 포트를 통과할 수 있는 트래픽 유형과 방향(수신/송신)을 지정할 수 있습니다. 보안 그룹 규칙은 상태 저장 L2-L4 트래픽 필터입니다.

6.8.9. L3 라우팅 및 NAT

OpenStack Networking 라우터는 여러 L2 네트워크를 연결할 수 있으며 하나 이상의 사설 L2 네트워크를 공유 외부 네트워크에 연결하는 게이트웨이(예: 인터넷에 액세스할 수 있는 공용 네트워크)를 제공할 수도 있습니다.

L3 라우터는 라우터를 외부 네트워크에 업링크하는 게이트웨이 포트에서 기본 SNAT(Network Address Translation) 기능을 제공합니다. 이 라우터 SNAT(소스 NAT)는 기본적으로 모든 송신 트래픽을 지원하며 유동 IP를 지원합니다. 이 IP는 외부 네트워크의 공용 IP에서 라우터에 연결된 다른 서브넷의 개인 IP로의 정적 일대일 양방향 매핑을 생성합니다. 유동 IP(DNAT를 통해)는 인스턴스에 대한 외부 인바운드 연결을 제공하며 한 인스턴스에서 다른 인스턴스로 이동할 수 있습니다.

프로젝트당 L3 라우팅과 Floating IP를 사용하여 프로젝트 인스턴스를 보다 세부적으로 연결해 보십시오. 공용 네트워크에 연결되거나 유동 IP를 사용하여 특별히 고려해야 합니다. 신중하게 고려한 보안 그

룹을 사용하면 외부에 노출되어야 하는 서비스에 대한 액세스를 필터링하는 것이 좋습니다.

6.8.10. QoS (Quality of Service)

기본적으로 **QoS(Quality of Service)** 정책 및 규칙은 클라우드 관리자가 관리하므로 프로젝트에서 특정 **QoS** 규칙을 만들거나 포트에 비정형 정책을 연결할 수 없게 됩니다. 일부 통신 애플리케이션과 같은 일부 사용 사례에서는 관리자가 프로젝트를 신뢰하여 **ir own** 정책을 만들고 포트에 연결할 수 있습니다. 이 작업은 **policy.json** 파일을 수정하여 수행할 수 있습니다.

Red Hat OpenStack Platform 12에서 **neutron**은 수신 및 송신 트래픽에 대한 대역폭 제한 **QoS** 규칙을 지원합니다. 이 **QoS** 규칙은 **QosBandwidthLimitRule** 로 명명되며 초당 킬로비트로 측정되는 **2**개의 부정적 정수를 허용합니다.

- **max-kbps: bandwidth**
- **max-burst-kbps: burst buffer**

QoSBandwidthLimitRule 은 **neutron Open vSwitch, Linux** 브리지 및 **SR-IOV** 드라이버에서 구현되었습니다. 그러나 **SR-IOV** 드라이버의 경우 **max-burst-kbps** 값이 사용되지 않으며 설정된 경우 무시됩니다.

QoS 규칙 **QosDscp MarkingRule** 은 **IPv4(RFC 2474)**의 서비스 헤더 유형과 모든 트래픽에서 **IPv6**의 트래픽 클래스 헤더를 사용하여 규칙이 적용되는 가상 머신에서 차별화된 서비스 헤더**(DSCP)** 값을 설정합니다. 이것은 네트워크를 통과할 때 패킷의 드롭 우선 순위를 나타내는 **21**개의 유효한 값이 있는 **6**비트 헤더입니다. 방화벽이 올바른 트래픽 또는 유효하지 않은 트래픽과 해당 액세스 제어 목록과 일치하도록 사용할 수도 있습니다.

6.8.11. 로드 밸런싱

OpenStack 로드 밸런싱 서비스**(octavia)**는 **Red Hat OpenStack Platform director** 설치를 위한 **LBaaS**(서비스로서의 부하 분산) 구현을 제공합니다. 로드 밸런싱을 달성하기 위해 **octavia**는 여러 공급업체 드라이버 활성화를 지원합니다. 참조 공급자 드라이버**(Amphora** 공급자 드라이버**)**는 오픈 소스, 확장 가능하고 가용성이 높은 로드 밸런싱 공급자입니다. 가상 머신 한 대를 관리하여 부하 분산 서비스를 제공합니다.총체적으로 **amphorae-**온디맨드로 구동됩니다.

로드 밸런싱 서비스에 대한 자세한 내용은 **Using Octavia for Load Balancing-as-a-Service** 가이드를 참조하십시오.

6.8.12. 네트워킹 서비스 강화

이 섹션에서는 **OpenStack** 배포 내의 프로젝트 네트워크 보안에 적용할 때 **OpenStack Networking** 구성 모범 사례를 설명합니다.

6.8.13. API 서버의 바인드 주소 제한: neutron-server

OpenStack Networking API 서비스가 들어오는 클라이언트 연결을 위해 네트워크 소켓을 바인딩하는 인터페이스 또는 **IP** 주소를 제한하려면 **/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf** 파일에 **bind_host** 및 **bind_por t** 를 지정합니다.

```
# Address to bind the API server
bind_host = IP ADDRESS OF SERVER

# Port the bind the API server to
bind_port = 9696
```

6.8.14. OpenStack Networking 서비스의 DB 및 RPC 통신 제한

OpenStack Networking 서비스의 다양한 구성 요소는 메시징 대기열 또는 데이터베이스 연결을 사용하여 **OpenStack Networking**의 다른 구성 요소와 통신합니다.

참고

RPC 통신이 필요한 모든 구성 요소에 대해 **15.3**절. “대기열 인증 및 액세스 제어” 에 제공된 지침을 따르는 것이 좋습니다.

6.8.15. 프로젝트 네트워크 서비스 워크플로

OpenStack Networking은 사용자에게 네트워크 리소스의 셀프 서비스 구성을 제공합니다. 클라우드 아키텍트와 운영자는 사용자에게 사용 가능한 네트워크 리소스를 생성, 업데이트 및 삭제할 수 있는 기능을 제공하는 데 있어 설계 활용 사례를 평가하는 것이 중요합니다.

6.8.16. 네트워킹 리소스 정책 엔진

OpenStack 네트워킹 내의 정책 엔진 및 해당 구성 파일(**policy.json**)은 프로젝트 네트워킹 방법 및 오브젝트에 대한 사용자의 더 세분화된 권한 부여를 제공하는 방법을 제공합니다. **OpenStack N** 외작업 정책 정의는 네트워크 가용성, 네트워크 보안 및 전반적인 **OpenStack** 보안에 영향을 미칩니다. 클라우드 아키텍트와 운영자는 네트워크 리소스 관리에 대한 사용자 및 프로젝트 액세스 권한을 신중하게 평가해야 합니다.

참고

보안 수준에 맞게 이 정책을 수정할 수 있으므로 기본 네트워킹 리소스 정책을 검토하는 것이 중요합니다.

OpenStack 배포에서 여러 다른 보안 영역에 여러 외부 액세스 지점을 제공하는 경우 프로젝트의 여러 **vNIC**를 여러 외부 액세스 **po intsTI**에 연결하는 기능을 제한하는 것이 중요합니다. 이 기능은 이러한 보안 영역을 브리지하고 예기치 않은 보안 손상으로 이어질 수 있습니다. **Compute(**컴퓨팅**)**에서 제공하는 호스트 집계 기능을 사용하거나 **e** 프로젝트 인스턴스를 서로 다른 가상 네트워크 구성으로 여러 프로젝트로 분할하여 이러한 위험을 완화할 수 있습니다. 호스트 집계에 대한 자세한 내용은 **https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html/configuring_the_compute_service_for_instance_creation/crea ting-and-managing-host-aggregates[**호스트 집계 및 관리**]** 링크를 참조하십시오.

6.8.17. 보안 그룹

보안 그룹은 보안 그룹 규칙의 컬렉션입니다. 보안 그룹 및 해당 규칙을 사용하면 관리자와 프로젝트에서 **ed**가 가상 인터페이스 포트를 통과할 수 있는 트래픽 유형과 방향(수신/송신)을 지정할 수 있습니다. **OpenStack Networking**에서 가상 인터페이스 포트를 만들면 보안 그룹과 연결됩니다. 배포별로 동작을 변경하기 위해 규칙을 **default** 보안 그룹에 추가할 수 있습니다.

계산 **API**를 사용하여 보안 그룹을 수정하는 경우 업데이트된 보안 그룹은 인스턴스의 모든 가상 인터페이스 포트에 적용됩니다. 이는 **neutron**에서 볼 수 있듯이 계산 보안 그룹 **API**가 포트 기반이 아닌 인스턴스 기반이기 때문입니다.

6.8.18. 할당량

할당량은 프로젝트에 사용할 수 있는 네트워크 리소스 수를 제한하는 기능을 제공합니다. 모든 프로젝트에 기본 할당량을 적용할 수 있습니다. 할당량 옵션을 검토하려면 **/var/lib/config-data/puppet-generated /neutron/etc/neutron/neutron.conf** 를 참조하십시오.

OpenStack Networking은 할당량 확장 **API**를 통해 프로젝트별 할당량 제한도 지원합니다. 프로젝트별 할당량을 사용하려면 **/var/lib/config-data/puppet-generated/neutron/e tc/neutron/neutron.conf** 에 **quota _driver** 옵션을 설정해야 합니다. 예를 들면 다음과 같습니다.

```
quota_driver = neutron.db.quota_db.DbQuotaDriver
```

6.8.19. ARP 스푸핑 완화

OpenStack Networking에는 인스턴스에 대한 **ARP** 스푸핑의 위협을 완화하는 데 도움이 되는 기능이 내장되어 있습니다. 결과적으로 발생하는 위험에 대해 신중하게 고려하지 않는 한 이 작업을 비활성화해

서는 안 됩니다.

6.8.20. config 파일의 사용자/그룹 소유권을 root/neutron으로 설정

구성 파일에는 구성 요소의 원활한 작동에 필요한 중요한 매개 변수 및 정보가 포함되어 있습니다. 권한이 없는 사용자가 의도하거나 실수로 매개변수 또는 파일 자체를 수정하거나 삭제하는 경우 심각한 가용성 문제로 인해 다른 최종 사용자에게 서비스 거부가 발생합니다. 따라서 이러한 중요한 구성 파일의 사용자 소유권을 root로 설정하고 그룹 소유권을 neutron으로 설정해야 합니다.

호스트에 neutron 그룹이 없습니다. *ls -l* 을 사용하면 그룹 소유자의 GID가 표시됩니다.

```
sudo ls -l /var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf
-rw-r-----. 1 root 42435 41748 Dec 11 09:34 /var/lib/config-data/puppet-
generated/neutron/etc/neutron/neutron.conf

sudo stat -L -c "%U %G" /var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf
root UNKNOWN
```

neuton.conf 구성 파일의 사용자 및 그룹 소유권이 각각 root 및 neutron 으로 설정되어 있는지 확인합니다.

```
sudo docker exec -it neutron_api stat -L -c "%U %G" /etc/neutron/neutron.conf
root neutron
```

6.8.21. 구성 파일의 엄격한 권한 설정

neutron.conf 설정 파일의 권한이 640 으로 설정되었는지 확인합니다.

```
$ stat -L -c "%a" /var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf
```

6.8.22. 인증에 Keystone 사용

/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf 에서 [DEFAULT] 섹션 아래의 auth_strategy 값이 keystone 으로 설정되었고 noauth 또는 noauth 2 가 아닌지 확인합니다.

6.8.23. 인증에 보안 프로토콜 사용

/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf 에서 [keystone _authtoken] 섹션 아래의 auth_ uri 값이 https로 시작하는 ID API 끝점으로 설정되어 있는지 확인합니

다.

6.8.24. Neutron API 서버에서 TLS 활성화

/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf 에서 [DEFAULT] 섹션 아래의 매개변수 use_ssl 이 True 로 설정되어 있는지 확인합니다.

7장. 네트워크 시간 프로토콜

Red Hat OpenStack Platform 클러스터 내의 시스템이 시스템 간에 정확하고 일관된 타임스탬프가 있는지 확인해야 합니다.

7.1. 일관된 시간이 중요한 이유

운영 및 보안 요구 사항 모두에서 조직 전체에서 일관된 시간이 중요합니다.

보안 이벤트 식별

일관된 시간 초과를 사용하면 이벤트 시퀀스를 이해할 수 있도록 영향을 받는 시스템에서 이벤트의 타임스탬프를 서로 연결할 수 있습니다.

인증 및 보안 시스템

보안 시스템은 시간 오차에 민감할 수 있습니다. 예를 들면 다음과 같습니다.

- **kerberos** 기반 인증 시스템은 클럭 스큐의 몇 초 동안 영향을 받는 클라이언트 인증을 거부할 수 있습니다.
- **TLS(Transport Layer Security)** 인증서는 유효한 시간 소스에 따라 다릅니다. 클라이언트와 서버 시스템 시간이 ***Valid From*** **date** 범위를 초과하면 클라이언트와 서버 **TLS** 연결에 실패합니다.

Red Hat OpenStack Platform 서비스

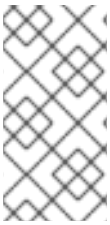
일부 핵심 **OpenStack** 서비스는 특히 **HA(**고가용성**)** 및 **Ceph**를 포함한 정확한 시간 유지에 따라 달라집니다.

7.2. NTP 설계

NTP(Network Time Protocol)는 계층 구조 설계로 구성됩니다. 각 계층을 계층이라고 합니다. 계층 구조의 맨 위에는 원자 시계와 같은 계층 **0** 장치가 있습니다. **NTP** 계층 구조에서 계층 **0** 장치는 공개적으로 사용 가능한 **stratum 1** 및 **stratum 2 NTP** 시간 서버에 대한 참조를 제공합니다.

데이터 센터 클라이언트를 공개적으로 사용 가능한 **NTP** 계층 **1** 또는 **2** 서버에 직접 연결하지 마십시오. 직접 연결 수를 사용하면 공용 **NTP** 리소스에 불필요한 부담이 발생할 수 있습니다. 대신 데이터 센터의 전용 시간 서버를 할당하고 해당 전용 서버에 클라이언트를 연결합니다.

호스트가 상주하는 호스트가 아닌 전용 시간 서버에서 시간을 받도록 인스턴스를 구성합니다.



참고

Red Hat OpenStack Platform 환경에서 실행되는 서비스 컨테이너는 여전히 호스트가 상주하는 호스트에서 시간을 받습니다.

8장. DIRECTOR를 사용하여 보안 강화 설정

Red Hat OpenStack Platform director를 사용하여 배포 프로세스의 일부로 보안 강화 값을 적용합니다. OpenStack 구성 파일은 director에서 관리하며 직접 편집하는 경우 덮어쓸 수 있습니다.

openstack overcloud deploy 를 실행할 때 변경 사항 외에 오버클라우드 배포에 필요한 모든 환경 파일을 항상 포함해야 합니다.

8.1. SSH 배너 텍스트 사용

SSH를 통해 연결하는 모든 사용자에게 콘솔 메시지를 표시하는 배너를 설정할 수 있습니다. 환경 파일에서 다음 매개 변수를 사용하여 배너 텍스트를 /etc/issue 에 추가할 수 있습니다. 요구 사항에 맞게 이 샘플 텍스트를 사용자 정의하는 것이 좋습니다.

```
resource_registry:
  OS::TripleO::Services::Sshd: ../puppet/services/sshd.yaml

parameter_defaults:
  BannerText: |
   ******************************************************************
   * This system is for the use of authorized users only. Usage of  *
   * this system may be monitored and recorded by system personnel. *
   * Anyone using this system expressly consents to such monitoring *
   * and is advised that if such monitoring reveals possible        *
   * evidence of criminal activity, system personnel may provide    *
   * the evidence from such monitoring to law enforcement officials.*
   ******************************************************************
```

이 변경 사항을 배포에 적용하려면 설정을 ssh_banner.yaml 이라는 파일로 저장한 다음 다음과 같이 오버클라우드 배포 명령에 전달합니다. <full environment> 는 원래의 모든 배포 매개변수를 계속 포함해야 함을 나타냅니다. 예를 들면 다음과 같습니다.

```
openstack overcloud deploy --templates \
  -e <full environment> -e  ssh_banner.yaml
```

8.2. 시스템 이벤트 감사

모든 감사 이벤트 레코드를 유지 관리하면 시스템 기준을 설정하거나, 문제 해결을 수행하거나, 특정 결과로 이어지는 이벤트 시퀀스를 분석할 수 있습니다. 감사 시스템은 시스템 시간 변경, 필수/공개 액세스 제어 변경, 사용자 또는 그룹 생성/삭제/삭제와 같은 다양한 유형의 이벤트를 로깅할 수 있습니다.

환경 파일을 사용하여 규칙을 생성할 수 있으며, 이 파일은 director에서 /etc/audit/audit.rules 에 삽입

합니다. 예를 들면 다음과 같습니다.

```
  resource_registry:
    OS::TripleO::Services::Auditd: /usr/share/openstack-tripleo-heat-
templates/puppet/services/auditd.yaml
  parameter_defaults:
    AuditdRules:
      'Record Events that Modify User/Group Information':
        content: '-w /etc/group -p wa -k audit_rules_usergroup_modification'
        order  : 1
      'Collects System Administrator Actions':
        content: '-w /etc/sudoers -p wa -k actions'
        order  : 2
      'Record Events that Modify the Systems Mandatory Access Controls':
        content: '-w /etc/selinux/ -p wa -k MAC-policy'
        order  : 3
```

8.3. 방화벽 규칙 관리

방화벽 규칙은 배포 중에 **Overcloud** 노드에 자동으로 적용되며 **OpenStack** 작동을 얻는 데 필요한 포트만 노출하기 위한 것입니다. 필요에 따라 추가 방화벽 규칙을 지정할 수 있습니다. 예를 들어 **Zabbix** 모니터링 시스템에 대한 규칙을 추가하려면 다음을 수행합니다.

```
parameter_defaults:
  ControllerExtraConfig:
    tripleo::firewall::firewall_rules:
      '301 allow zabbix':
        dport: 10050
        proto: tcp
        source: 10.0.0.8
        action: accept
```

액세스를 제한하는 규칙을 추가할 수도 있습니다. 규칙 정의 중에 사용되는 숫자는 규칙의 우선 순위를 결정합니다. 예를 들어 **RabbitMQ**의 규칙 번호는 기본적으로 **109** 입니다. 이 문제를 완화하려면 더 낮은 값을 사용하도록 전환합니다.

```
parameter_defaults:
  ControllerExtraConfig:
    tripleo::firewall::firewall_rules:
      '098 allow rabbit from internalapi network':
        dport: [4369,5672,25672]
        proto: tcp
        source: 10.0.0.0/24
        action: accept
      '099 drop other rabbit access':
        dport: [4369,5672,25672]
        proto: tcp
        action: drop
```

이 예에서 **098** 및 **099** 는 **RabbitMQ**의 규칙 번호 **109** 보다 낮은 임의로 선택된 숫자입니다. 규칙 번호를 확인하기 위해 적절한 노드에서 **iptables** 규칙을 검사할 수 있습니다. **RabbitMQ**의 경우 컨트롤러를 확인합니다.

```
iptables-save
[...]
-A INPUT -p tcp -m multiport --dports 4369,5672,25672 -m comment --comment "109 rabbitmq" -m
state --state NEW -j ACCEPT
```

또는 **puppet** 정의에서 포트 요구 사항을 추출할 수 있습니다. 예를 들어 **RabbitMQ**의 규칙은 **puppet/services/rabbitmq.yaml**에 저장됩니다.

```
tripleo.rabbitmq.firewall_rules:
  '109 rabbitmq':
    dport:
      - 4369
      - 5672
      - 25672
```

규칙에 대해 다음 매개변수를 설정할 수 있습니다.

- 포트: 규칙에 연결된 포트입니다. **puppetlabs-firewall** 에서 사용하지 않음.
- **dport:** 규칙에 연결된 대상 포트입니다.
- **sport:** 규칙과 연결된 소스 포트입니다.
- **proto:** 규칙과 연결된 프로토콜입니다. 기본값은 ***tcp***입니다.
- 작업: 규칙과 연결된 작업 정책입니다. 기본값은 ***accept***입니다
- **jump:** 로 이동할 체인입니다.
- **state:** 규칙과 관련된 상태 배열입니다. 기본값 **[*NEW*]**

- **source:** 규칙과 연결된 소스 **IP** 주소입니다.
- **iniface:** 규칙에 연결된 네트워크 인터페이스입니다.
- 체인**:** 규칙과 연결된 체인입니다. 기본적으로 ***INPUT***로 설정됩니다.
- 대상**:** 규칙에 연결된 대상 **cidr**입니다.
- 추가 내용**: puppetlabs-firewall** 모듈에서 지원하는 추가 매개 변수의 해시입니다.

8.4. AIDE를 사용한 침입 탐지

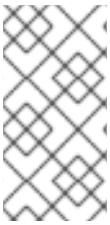
AIDE(고급 침입 감지 환경**)**는 파일 및 디렉토리 무결성 검사기입니다. 이는 권한 없는 파일 조작 또는 변경의 사고를 감지하는 데 사용됩니다. 예를 들어 **AIDE**에서 시스템 암호 파일이 변경된 경우 경고를 표시할 수 있습니다.

AIDE는 시스템 파일을 분석한 다음 파일 해시의 무결성 데이터베이스를 컴파일하여 작동합니다. 그런 다음 데이터베이스는 파일 및 디렉터리의 무결성을 확인하고 변경 사항을 탐지하는 비교 지점 역할을 합니다.

director에는 **AIDE** 서비스가 포함되어 있으므로 **AIDE** 구성에 항목을 추가하면 **AIDE** 서비스에서 무결성 데이터베이스를 생성할 수 있습니다. 예를 들면 다음과 같습니다.

```
resource_registry:
  OS::TripleO::Services::Aide: ../puppet/services/aide.yaml

parameter_defaults:
  AideRules:
    'TripleORules':
      content: 'TripleORules = p+sha256'
      order: 1
    'etc':
      content: '/etc/ TripleORules'
      order: 2
    'boot':
      content: '/boot/ TripleORules'
      order: 3
    'sbin':
      content: '/sbin/ TripleORules'
      order: 4
```

```
'var':
  content: '/var/ TripleORules'
  order: 5
'not var/log':
  content: '!/var/log.*'
  order: 6
'not var/spool':
  content: '!/var/spool.*'
  order: 7
'not nova instances':
  content: '!/var/lib/nova/instances.*'
  order: 8
```



참고

위의 예는 적극적으로 유지 관리되거나 벤치마킹되어 있지 않으므로 요구 사항에 맞는 **AIDE** 값을 선택해야 합니다.

1. **TripleORules** 라는 별칭은 매번 동일한 특성을 반복해서 사용하지 않도록 선언됩니다.

2. 별칭은 **p+sha256** 의 특성을 받습니다. **AIDE** 용어에서는 **sha256** 의 무결성 체크섬을 사용하여 모든 파일 권한 **p** 를 모니터링합니다.

AIDE의 구성 파일에 사용할 수 있는 속성의 전체 목록은 **https://aide.github.io/** 의 **AIDE MAN** 페이지를 참조하십시오.

배포에 변경 사항을 적용하려면 다음을 완료합니다.

1. 설정을 **/home/stack/templates/** 디렉터리에 **aide.yaml** 이라는 파일로 저장합니다.

2. **OS::TripleO::Services::Aide** 매개변수의 값이 상대 경로에서 절대 경로가 되도록 변경해야 합니다.

```
OS::TripleO::Services::Aide: /usr/share/openstack-tripleo-heat-
templates/puppet/services/aide.yaml
```

3. 환경에 고유한 기타 모든 필수 **Heat** 템플릿 및 환경 파일과 함께 **openstack overcloud deploy** 명령에 **/home/stack/templates/aide.yaml** 환경 파일을 포함합니다.

```
openstack overcloud deploy --templates
...
-e /home/stack/templates/aide.yaml
```

8.4.1. 복잡한 AIDE 규칙 사용

이전에 설명한 형식을 사용하여 복잡한 규칙을 생성할 수 있습니다. 예를 들면 다음과 같습니다.

```
MyAlias = p+i+n+u+g+s+b+m+c+sha512
```

위 명령은 체크섬 생성에 **sha256**을 사용하여 권한, **inode**, 링크 수, 사용자, 그룹, 크기, 블록 수, **mtime, ctime**을 사용하는 명령과 같이 변환됩니다.

별칭은 항상 **1** 의 순서 위치가 있어야 합니다. 즉 **AIDE** 규칙의 맨 위에 있고 아래의 모든 값에 재귀적으로 적용됩니다.

별칭 뒤에는 모니터링할 디렉터리가 있습니다. 정규 표현식을 사용할 수 있습니다. 예를 들어 **var** 디렉토리에 대한 모니터링을 설정하지만 **!** 를 **'!/var/ log.*'** 및 **'!/var/spool.*'** 과 함께 사용하여 **not** 절으로 덮어씁니다.

8.4.2. 추가 AIDE 값

다음 **AIDE** 값을 사용할 수도 있습니다.

AideConfPath: aide 구성 파일의 전체 **POSIX** 경로이며 기본값은 **/etc/aide.conf** 입니다. 파일 위치를 변경할 필요가 없는 경우 기본 경로를 유지하는 것이 좋습니다.

AideDBPath: AIDE 무결성 데이터베이스에 대한 전체 **POSIX** 경로입니다. **AIDE** 데이터베이스 파일이 읽기 전용 파일 마운트에 노드에서 저장되는 경우가 많으므로 이 값은 운영자가 자체 전체 경로를 선언할 수 있도록 구성할 수 있습니다.

AideDBTempPath: AIDE 무결성 임시 데이터베이스에 대한 전체 **POSIX** 경로입니다. 이 임시 파일은 **AIDE**에서 새 데이터베이스를 초기화할 때 생성됩니다.

AideHour: 이 값은 **AIDE cron** 구성의 일부로 **hour** 속성을 설정하는 것입니다.

AideMinute: 이 값은 **AIDE cron** 구성의 일부로 **minute** 속성을 설정하는 것입니다.

AideCronUser: 이 값은 **linux** 사용자를 **AIDE cron** 구성의 일부로 설정하는 것입니다.

AideEmail: 이 값은 **cron** 실행이 수행될 때마다 **AIDE**에서 보고하는 이메일 주소를 설정합니다.

AideMuaPath: 이 값은 **AIDE** 보고서를 **AideEmail** 내에 설정된 이메일 주소로 보내는 데 사용되는 메일 사용자 에이전트의 경로를 설정합니다.

8.4.3. AIDE의 Cron 구성

AIDE director 서비스를 사용하면 **cron** 작업을 구성할 수 있습니다. 기본적으로 이메일 경고를 사용하려면 **/var/log/audit/;**로 보고서를 전송한 다음 **AideEmail** 매개 변수를 활성화하여 구성된 이메일 주소로 경고를 보냅니다. 중요한 경고에 대한 이메일 의존은 시스템 중단 및 의도하지 않은 메시지 필터링에 취약할 수 있습니다.

8.4.4. 시스템 업그레이드 효과 고려

업그레이드가 수행되면 **AIDE** 서비스는 새 무결성 데이터베이스를 자동으로 다시 생성하여 업그레이드된 모든 파일을 올바르게 다시 계산하여 업데이트된 체크섬을 보유합니다.

openstack overcloud deploy 를 초기 배포에 대한 후속 실행으로 호출하고 **AIDE** 구성 규칙이 변경되면 **director AIDE** 서비스에서 데이터베이스를 다시 빌드하여 새 구성 특성이 무결성 데이터베이스에 캡슐화되도록 합니다.

8.5. SECURETTY 검토

securetty를 사용하면 모든 콘솔 장치**(tty)**에 대한 루트 액세스를 비활성화할 수 있습니다. 이 동작은 **/etc/securetty** 파일의 항목으로 관리합니다. 예를 들면 다음과 같습니다.

```
resource_registry:
  OS::TripleO::Services::Securetty: ../puppet/services/securetty.yaml

parameter_defaults:
  TtyValues:
    - console
    - tty1
    - tty2
    - tty3
```

```
    - tty4
    - tty5
    - tty6
```

8.6. 아이덴티티 서비스용 CADF 감사

철저한 감사 프로세스를 통해 **OpenStack** 배포의 지속적인 보안 포스처를 검토하는 데 도움이 될 수 있습니다. 이는 보안 모델에서 역할 때문에 **keystone**에 특히 중요합니다.

Red Hat OpenStack Platform은 감사 이벤트를 위한 데이터 형식으로 **Cloud Auditing Data Federation(CADF)**을 채택했으며, **keystone** 서비스에서 **ID** 및 토큰 작업을 위한 **CADF** 이벤트를 생성했습니다. **KeystoneNotificationFormat** 을 사용하여 **keystone**에 대한 **CADF** 감사를 활성화할 수 있습니다.

```
parameter_defaults:
  KeystoneNotificationFormat: cadf
```

8.7. LOGIN.DEFS 값 확인

새 시스템 사용자(**keystone**이 아닌)에 대한 암호 요구 사항을 적용하기 위해 다음 예제 매개 변수를 따라 **/etc/login.defs** 에 항목을 추가할 수 있습니다.

```
resource_registry:
  OS::TripleO::Services::LoginDefs: ../puppet/services/login-defs.yaml

parameter_defaults:
  PasswordMaxDays: 60
  PasswordMinDays: 1
  PasswordMinLen: 5
  PasswordWarnAge: 7
  FailDelay: 4
```

9장. 대시보드 서비스 강화

이 장에서는 OpenStack 대시보드(horizon)를 사용하는 Red Hat OpenStack Platform 배포의 보안 강화 고려 사항에 대해 설명합니다.

대시보드는 사용자에게 자체 리소스를 프로비저닝할 수 있는 셀프 서비스 포털을 제공합니다(관리자가 설정한 제한 내에서). 예를 들어 대시보드를 사용하여 인스턴스 플레이버를 정의하고, VM(가상 시스템) 이미지를 업로드하고, 가상 네트워크를 관리하고, 보안 그룹을 생성하고, 인스턴스를 시작하고, 관리 콘솔을 통해 인스턴스에 원격으로 액세스할 수 있습니다.

클라우드 리소스 및 구성에 대한 액세스 권한을 부여할 수 있으므로 대시보드는 OpenStack API와 동일한 민감도로 처리해야 합니다.

9.1. 계획 대시보드 배포

이 섹션에서는 대시보드(horizon) 서비스를 배포하기 전에 고려해야 할 보안 측면에 대해 설명합니다.

9.1.1. 도메인 이름 선택

대시보드를 공유 하위 도메인(예: https://openstack.example.org 또는 https://horizon.openstack.example.org)에 배포하는 대신 두 번째 수준 도메인(예: https://example.com)에 배포하는 것이 좋습니다. 또한 https://horizon/ 과 같은 베어 내부 도메인에 대시보드를 배포하지 않는 것이 좋습니다. 이러한 권장 사항은 same-origin-policy 브라우저의 제한 사항을 기반으로 합니다.

이 접근 방식을 사용하면 콘텐츠를 완전히 제어하지 못할 수 있는 다른 도메인에서 쿠키 및 보안 토큰을 격리할 수 있습니다. 하위 도메인에 배포되는 경우 대시보드의 보안은 동일한 두 번째 수준 도메인에 배포된 최소의 보안 애플리케이션과 동일합니다.

쿠키 지원 세션 저장소를 피하고 HTS(HTTP Strict Transport Security)(이 가이드에 설명됨)를 구성하여 이러한 위험을 추가로 완화할 수 있습니다.

9.1.2. ALLOWED_HOSTS 구성

웹 서비스는 거짓 HTTP 호스트 헤더와 연결된 위협에 취약할 수 있습니다. 이 문제를 완화하려면 OpenStack 대시보드에서 제공하는 FQDN을 사용하도록 ALLOWED_HOSTS 설정을 구성하는 것이 좋습니다.

구성되고 나면 들어오는 HTTP 요청의 Host: 헤더 값이 이 목록의 값과 일치하지 않으면 오류가 발생하고 요청자가 진행되지 않습니다.

Horizon은 python Django 웹 프레임워크를 기반으로 구축되어 잠재적으로 HTTP 호스트: 헤더를 조작하지 않도록 ALLOWED_HOSTS 를 설정해야 합니다. 이 값을 에서 대시보드에 액세스할 수 있어야 하는 FQDN으로 설정합니다. director의 경우 이 설정은 HorizonAllowedHosts에서 관리합니다.

9.2. 일반적인 웹 서버 취약점 이해

이 장에서는 몇 가지 일반적인 웹 서버 취약점을 완화하는 방법을 설명합니다.

9.2.1. 교차 사이트 스크립팅(XSS)

OpenStack 대시보드는 사용자 지정할 수 있으며 대부분의 필드에서 전체 유니코드 문자 세트를 허용합니다. 이러한 확장성을 통해 XSS(Cross-site Scripting) 취약점이 도입될 수 있습니다. Horizon에는 개발자가 의도치 않게 XSS 취약점을 생성하지 않도록 도와주는 도구가 포함되어 있지만 개발자가 올바르게 사용하는 경우에만 작동합니다. 사용자 지정 대시보드를 감사하고 다음 기능에 특히 주의하는 것이 좋습니다.

- mark_safe 함수.
- is_safe - 사용자 지정 템플릿 태그와 함께 사용하는 경우입니다.
- safe 템플릿 태그입니다.
- 어디에서나 자동 이스케이프가 해제되고 부적절하게 이스케이프된 데이터를 평가할 수 있는 JavaScript가 모두 해제됩니다.

9.2.2. cross Site Request Forgery (CSRF)

OpenStack 대시보드는 위협이 발생할 가능성이 있으므로 개발자가 사용자 지정 대시보드를 사용하여 사이트 간 스크립팅 취약점을 도입하지 않도록 설계되었습니다. 여러 JavaScript 인스턴스를 사용하는 대시보드는 @csrf_exempt 데코레이터를 부적절하게 사용하는 등 취약점에 대해 감사해야 합니다. 이러한 권장 보안 설정을 따르지 않는 대시보드는 CORS(Cross Origin Resource Sharing) 제한이 완화되기 전에 신중하게 평가해야 합니다.

대시보드 도메인 및 프로토콜만 허용하도록 각 응답이 포함된 제한적인 **CORS** 헤더를 보내도록 웹 서버를 구성해야 합니다. 예**:access-Control-Allow-Origin: https://example.com/.** 와일드카드 원점을 허용하지 않아야 합니다.

9.2.3. 크로스 프레임 스크립팅(XFS)

disallow _iframe_embed 설정은 대시보드가 **iframe** 내에 임베드되지 않도록 합니다. 기존 브라우저는 여전히 **XFS(Cross-frame Scripting)** 취약점에 취약할 수 있으므로, 이 옵션은 **iframe**이 필요하지 않은 배포에 추가로 보안 강화를 추가합니다.

다음 매개변수를 사용하여 **iframe** 임베딩을 허용할 수 있습니다.

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disallow_iframe_embed: false
```

9.2.4. 대시보드 트래픽에 HTTPS 암호화 사용

HTTPS를 사용하여 대시보드 트래픽을 암호화하는 것이 좋습니다. 공인 **CA(**인증 기관**)**에서 신뢰할 수 있는 유효한 인증서를 사용하도록 구성하여 이 작업을 수행할 수 있습니다. 개인 조직에서 발급한 인증서는 신뢰할 수 있는 루트가 모든 사용자 브라우저에 사전 설치된 경우에만 적합합니다.

대시보드 도메인에 대한 **HTTP** 요청을 구성하여 정규화된 **HTTPS URL**로 리디렉션합니다.

director 기반 배포의 경우 https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html/advanced_overcloud_customization/sect-enabling_ssltls_on_the_overcloud 을 참조하십시오.

9.2.5. HSTS(HTTP Strict Transport Security)

HSTS(HTTP Strict Transport Security)는 처음에 보안 연결을 만든 후 브라우저가 후속 비보안 연결을 수행하지 못하도록 합니다. 공용 또는 신뢰할 수 없는 영역에 **HTTP** 서비스를 배포한 경우 **HSTS**가 특히 중요합니다.

director 기반 배포의 경우 이 설정은 기본적으로 활성화되어 있습니다.

```
enable_secure_proxy_ssl_header: true
```

9.3. 대시보드 콘텐츠 캐싱

9.3.1. 프런트 엔드 캐싱

OpenStack API 요청에서 직접 발생하는 동적 콘텐츠를 렌더링하므로 대시보드에 프런트엔드 캐싱 툴을 사용하지 않는 것이 좋습니다. 결과적으로 **varnish** 와 같은 프론트엔드 캐싱 계층을 통해 올바른 콘텐츠가 표시되지 않을 수 있습니다. 대시보드는 웹 서비스에서 직접 제공하는 정적 미디어를 제공하는 **Django**를 사용하며 웹 호스트 캐싱의 이점을 이미 제공합니다.

9.3.2. 세션 백엔드

director 기반 배포의 경우 **Horizon**의 기본 세션 백엔드는 **memcached**와 결합된 **django.contrib.sessions.backends.cache** 입니다. 이 방법은 성능을 위해 로컬 메모리 캐시를 사용하는 것이 좋습니다. 고가용성 및 로드 밸런싱 설치의 경우 더 안전하며 여러 서버에서 캐시를 공유하는 동시에 계속 단일 캐시로 처리할 수 있습니다.

director의 **Horizon.yaml** 파일에서 이러한 설정을 검토할 수 있습니다.

```
horizon::cache_backend: django.core.cache.backends.memcached.MemcachedCache
horizon::django_session_engine: 'django.contrib.sessions.backends.cache'
```

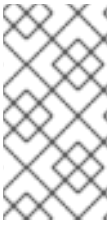
9.4. 비밀 키 검토

대시보드는 일부 보안 기능의 공유 **SECRET_KEY** 설정을 사용합니다. 비밀 키는 임의로 생성된 문자열 **64**자 이상이어야 하며, 모든 활성 대시보드 인스턴스에서 공유해야 합니다. 이 키를 손상시켜 원격 공격자가 임의의 코드를 실행할 수 있습니다. 이 키를 순환하면 기존 사용자 세션 및 캐싱이 무효화됩니다. 이 키를 공개 리포지토리에 커밋하지 마십시오.

director 배포의 경우 이 설정은 **HorizonSecret** 값으로 관리됩니다.

9.5. 세션 쿠키 구성

대시보드 세션 쿠키를 열어 **JavaScript**와 같은 브라우저 기술을 통해 상호 작용할 수 있습니다. 모든 곳에 **TLS**를 사용한 **director** 배포의 경우 **HorizonSecure** 설정으로 이 동작을 강화할 수 있습니다.



참고

선행 점이 있는 와일드카드 도메인을 사용하도록 **CSRF** 또는 세션 쿠키를 구성하지 마십시오.

9.6. 정적 미디어

대시보드의 정적 미디어를 대시보드 도메인의 하위 도메인에 배포하고 웹 서버에서 제공해야 합니다. 외부 **CDN**(콘텐츠 전달 네트워크)도 사용할 수 있습니다. 이 하위 도메인은 쿠키를 설정하거나 사용자 제공 콘텐츠를 제공하지 않아야 합니다. 미디어는 **HTTPS**와 함께 제공해야 합니다.

대시보드의 기본 구성에서는 **django_compressor** 를 사용하여 **CSS** 및 **JavaScript** 콘텐츠를 압축하고 축소한 후 서비스를 제공합니다. 이 프로세스는 기본 **in-request** 동적 압축을 사용하고 배포된 코드 또는 **CDN** 서버와 함께 결과 파일을 복사하는 대신 대시보드를 배포하기 전에 정적으로 수행해야 합니다. 압축은 프로덕션이 아닌 빌드 환경에서 수행해야 합니다. 이 방법이 실용적이지 않으면 리소스 압축을 완전히 비활성화하는 것이 좋습니다. 온라인 압축 종속성(기본, **Node.js**)은 프로덕션 시스템에 설치해서는 안 됩니다.

9.7. 암호 복잡성 검증

OpenStack Dashboard(horizon)는 암호 유효성 검사 검사를 사용하여 암호 복잡성을 적용할 수 있습니다.

암호 검증을 위한 정규식과 실패한 테스트에 대해 표시할 도움말 텍스트를 지정할 수 있습니다. 다음 예제에서는 사용자가 **8~18**자 길이의 암호를 생성해야 합니다.

```
parameter_defaults:
  HorizonPasswordValidator: '^.{8,18}$'
  HorizonPasswordValidatorHelp: 'Password must be between 8 and 18 characters.'
```

배포에 이 변경 사항을 적용하려면 설정을 **Horizon_password.yaml** 이라는 파일로 저장한 다음 다음과 같이 오버클라우드 배포 명령에 전달합니다. **<full environment>** 는 원래의 모든 배포 매개변수를 계속 포함해야 함을 나타냅니다. 예를 들면 다음과 같습니다.

```
openstack overcloud deploy --templates \
  -e <full environment> -e  horizon_password.yaml
```

9.8. 관리자 암호 검사 시행

다음 설정은 기본적으로 **True** 로 설정되지만 필요한 경우 환경 파일을 사용하여 비활성화할 수 있습니

다.

참고

이러한 설정은 보안에 미치는 잠재적 영향을 완전히 파악한 경우에만 **False** 로 설정해야 합니다.

대시보드의 **local _settings.py** 파일의 **ENFORCE_PASSWORD_ CHECK** 설정은 관리자가 *암호 변경을 시작하는지 확인하는 데 도움이 되는* ***Change Password (암호 변경) 양식에 Admin*** **Password(**관리 암호**)** 필드를 표시합니다.

환경 파일을 사용하여 **ENFORCE_PASSWORD_CHECK** 를 비활성화할 수 있습니다.

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::enforce_password_check: false
```

9.9. IFRAME 임베딩 허용

참고

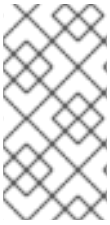
이러한 설정은 보안에 미치는 잠재적 영향을 완전히 파악한 경우에만 **False** 로 설정해야 합니다.

DISALLOW_IFRAME_EMBED 설정은 **iframe** 내에 포함된 대시보드를 허용하지 않습니다. 기존 브라우저는 여전히 **XFS(Cross-frame Scripting)** 취약점에 취약할 수 있으므로, 이 옵션은 **iframe**이 필요하지 않은 배포에 추가로 보안 강화를 추가합니다. 설정은 기본적으로 **True** 로 설정되지만 필요한 경우 환경 파일을 사용하여 비활성화할 수 있습니다. 예를 들어 다음 매개변수를 사용하여 **iframe** 포함을 허용할 수 있습니다.

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disallow_iframe_embed: false
```

9.10. 암호 표시 비활성화

다음 설정은 기본적으로 **True** 로 설정되지만 필요한 경우 환경 파일을 사용하여 비활성화할 수 있습니다.



참고

이러한 설정은 보안에 미치는 잠재적 영향을 완전히 파악한 경우에만 **False** 로 설정해야 합니다.

Password(암호) 단추를 누르면 대시보드의 사용자가 입력하려는 암호를 확인할 수 있습니다. 이 옵션은 **DISABLE_PASSWORD_REVEAL** 매개변수를 사용하여 전환할 수 있습니다.

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disable_password_reveal: false
```

9.11. 대시보드의 로그인 배너 표시

HIPAA, PCI-DSS, 미국 등의 규제 받는 산업. 정부, 사용자 로그온 배너를 표시해야 합니다. 대시보드 서비스는 컨테이너 내에서 실행되므로 배너를 적용하려면 대시보드 컨테이너 이미지를 사용자 지정해야 합니다. 대시보드 컨테이너 사용자 지정에 대한 자세한 내용은 https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html-single/introduction_to_the_openstack_dashboard/index#dashboard-customization 을 참조하십시오.

9.11.1. 테마 사용자 정의

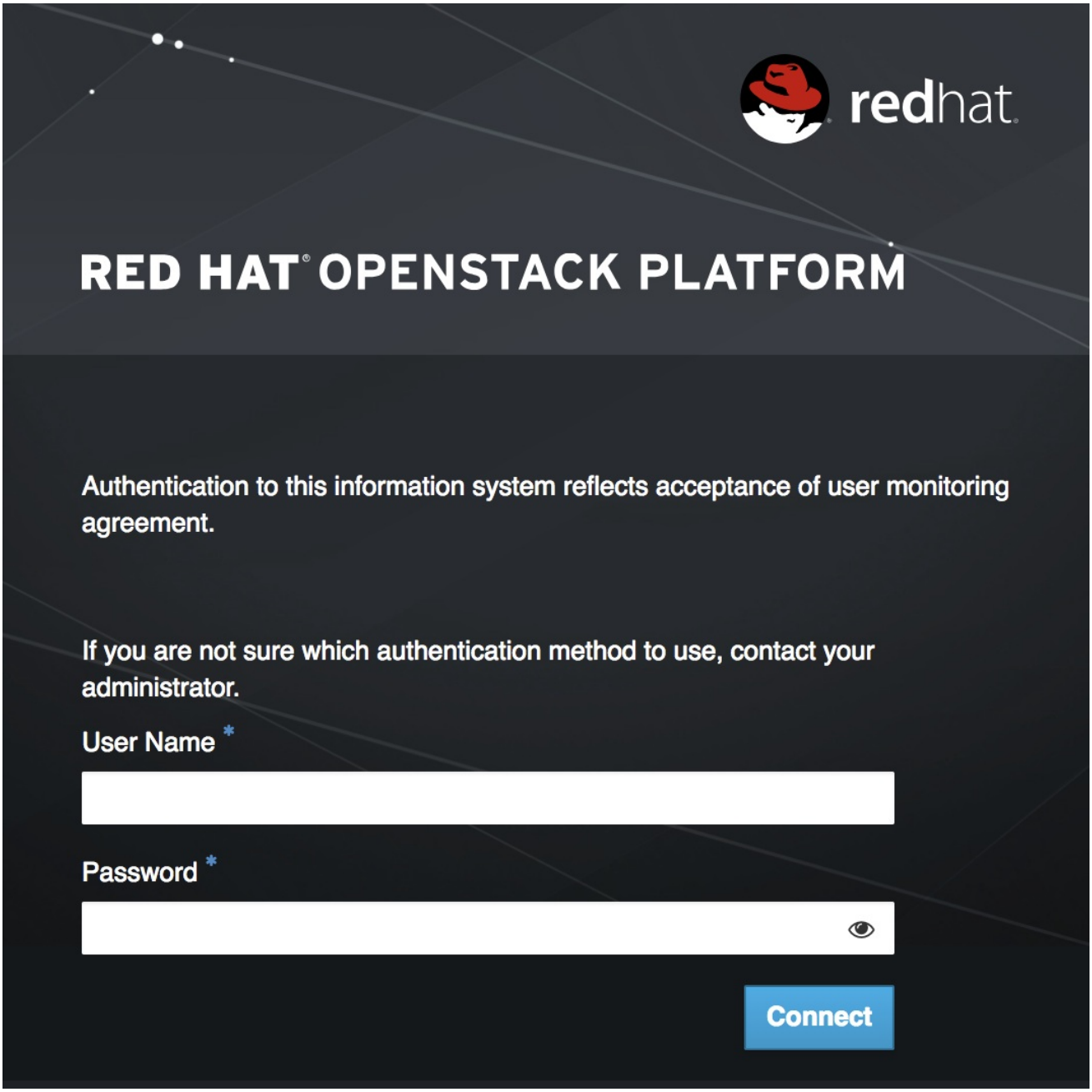
사용자 정의 대시보드 컨테이너에서 **/usr/share/openstack-dashboard/openstack_dashboard/openstack_dashboard/ groupss/rcue/templates/auth/login.html** 파일을 수동으로 편집하여 로그 배너를 생성할 수 있습니다.

1. **{% include 'auth/_login.html' %}** 섹션 바로 앞에 필요한 로그온 배너를 입력합니다. **HTML** 태그는 허용됩니다. 예를 들면 다음과 같습니다.

```
<snip>
<div class="container">
  <div class="row-fluid">
    <div class="span12">
      <div id="brand">
        <img src="../../static/themes/rcue/images/RHOSP-Login-Logo.svg">
      </div><!--/#brand-->
    </div><!--/.span*-->

    <!-- Start of Logon Banner -->
    <p>Authentication to this information system reflects acceptance of user monitoring
agreement.</p>
    <!-- End of Logon Banner -->
```

```
    {% include 'auth/_login.html' %}
  </div><!--/.row-fluid→
</div><!--/.container-->

{% block js %}
  {% include "horizon/_scripts.html" %}
{% endblock %}

  </body>
</html>
```

업데이트된 대시보드는 다음과 유사합니다.



9.12. 파일 업로드 크기 제한

선택적으로 파일 업로드 크기를 제한하도록 대시보드를 구성할 수 있습니다. 이 설정은 다양한 보안 강화 정책에 대한 요구 사항일 수 있습니다.

LimitRequestBody - 이 값(바이트)은 이미지 및 기타 대용량 파일과 같은 대시보드를 사용하여 전송할 수 있는 파일의 최대 크기를 제한합니다.

중요

이 설정은 **Red Hat**에서 공식적으로 테스트하지 않았습니다. 프로덕션 환경에 배포하기 전에 이 설정의 효과를 철저하게 테스트하는 것이 좋습니다.

참고

값이 너무 작으면 파일 업로드가 실패합니다.

예를 들어 이 설정은 각 파일이 최대 **10GB(10737418240)**로 제한합니다. 배포에 맞게 이 값을 조정해야 합니다.

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf/httpd.conf**

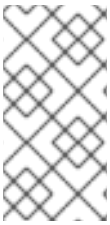
```
<Directory />
  LimitRequestBody 10737418240
</Directory>
```

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/10-horizon_vhost.conf**

```
<Directory "/var/www">
  LimitRequestBody 10737418240
</Directory>
```

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/15-horizon_ssl_vhost.conf**

```
<Directory "/var/www">
  LimitRequestBody 10737418240
</Directory>
```



참고

이러한 구성 파일은 **Puppet**에서 관리하므로 **openstack overcloud deploy** 프로세스를 실행할 때마다 관리되지 않는 변경 사항을 덮어씁니다.

9.13. 대시보드 서비스 디버깅

프로덕션 환경에서는 **DEBUG** 설정을 **False** 로 설정하는 것이 좋습니다. **True** 로 설정하면 **Django**에서 중요한 웹 서버 상태 정보가 포함된 브라우저 사용자에게 스택 추적을 출력할 수 있습니다. 또한 **DEBUG** 모드는 요구 사항에 따라 바람직하지 않은 결과일 수 있는 **ALLOWED_HOSTS** 를 비활성화하는 효과가 있습니다.

10장. 공유 파일 시스템 강화(MANILA)

Shared File System 서비스(**manila**)는 다중 프로젝트 클라우드 환경에서 공유 파일 시스템을 관리하는 일련의 서비스를 제공합니다. **OpenStack**에서 **Block Storage** 서비스(**cinder**) 프로젝트를 통해 블록 기반 스토리지 관리를 제공하는 방법과 유사합니다. **manila**를 사용하면 공유 파일 시스템을 생성하고 가시성, 접근성 및 사용량 할당량과 같은 해당 속성을 관리할 수 있습니다.

manila에 대한 자세한 내용은 스토리지 가이드 https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html-single/storage_guide/를 참조하십시오.

10.1. MANILA에 대한 보안 고려 사항

Manila는 **keystone**에 등록되어 있으므로 **manila endpoints** 명령을 사용하여 **API**를 찾을 수 있습니다. 예를 들면 다음과 같습니다.

```
$ manila endpoints
+-------------+-----------------------------------------+
| manila      | Value                                   |
+-------------+-----------------------------------------+
| adminURL    | http://172.18.198.55:8786/v1/20787a7b...|
| region      | RegionOne                               |
| publicURL   | http://172.18.198.55:8786/v1/20787a7b...|
| internalURL | http://172.18.198.55:8786/v1/20787a7b...|
| id          | 82cc5535aa444632b64585f138cb9b61        |
+-------------+-----------------------------------------+

+-------------+-----------------------------------------+
| manilav2    | Value                                   |
+-------------+-----------------------------------------+
| adminURL    | http://172.18.198.55:8786/v2/20787a7b...|
| region      | RegionOne                               |
| publicURL   | http://172.18.198.55:8786/v2/20787a7b...|
| internalURL | http://172.18.198.55:8786/v2/20787a7b...|
| id          | 2e8591bfcac4405fa7e5dc3fd61a2b85        |
+-------------+-----------------------------------------+
```

기본적으로 **manila API** 서비스는 **IPv4** 및 **IPv6**을 모두 지원하는 **tcp6** 이 있는 포트 **8786** 에서만 수신 대기합니다.

Manila는 여러 구성 파일을 사용합니다. 이러한 파일은 **/var/lib/config-data/puppet-generated/manila/** 에 저장됩니다.

```
api-paste.ini
manila.conf
```

```
policy.json
rootwrap.conf
rootwrap.d

./rootwrap.d:
share.filters
```

루트가 아닌 서비스 계정에서 실행되도록 **manila**를 구성하고 시스템 관리자만 수정할 수 있도록 파일 권한을 변경하는 것이 좋습니다. **Manila**에서는 관리자만 구성 파일에 쓸 수 있으며 서비스는 **manila** 그룹의 그룹 멤버십을 통해서만 읽을 수 있습니다. 다른 사용자는 서비스 계정 암호가 포함되어 있으므로 이러한 파일을 읽을 수 없어야 합니다.

참고

root 사용자만 **rootwrap .conf**의 **manila-rootwrap** 의 구성에 쓸 수 있어야 하며 **rootwrap. d/share.filters**의 공유 노드에 대한 **manila-rootwrap** 명령 필터가 있어야 합니다.

10.2. MANILA의 네트워크 및 보안 모델

manila의 공유 드라이버는 백엔드에 대해 공급업체별 공유 작업을 관리할 수 있는 **Python** 클래스입니다. 백엔드는 **manila-share** 서비스의 인스턴스입니다. **Manila**는 상용 벤더와 오픈 소스 솔루션을 모두 지원하는 다양한 스토리지 시스템용 드라이버를 공유합니다. 각 공유 드라이버는 공유 서버 및 공유 서버가 없는 하나 이상의 백엔드 모드를 지원합니다. 관리자는 **driver_handles_share_servers** 를 사용하여 **manila.conf** 에 지정하여 모드를 선택합니다.

공유 서버는 공유 파일 시스템을 내보내는 논리적 **NAS(**네트워크 연결 스토리지**)** 서버입니다. 오늘날 백엔드 스토리지 시스템은 정교하며 다양한 **OpenStack** 프로젝트 간에 데이터 경로와 네트워크 경로를 격리할 수 있습니다.

manila 공유 드라이버에서 프로비저닝한 공유 서버는 이를 생성하는 프로젝트 사용자에 속하는 격리된 네트워크에 생성됩니다. 공유 서버 모드는 네트워크 프로바이더에 따라 플랫 네트워크 또는 세그먼트화된 네트워크로 구성할 수 있습니다.

동일한 하드웨어를 사용하는 다양한 모드에 대해 별도의 드라이버를 사용할 수 있습니다. 선택한 모드에 따라 구성 파일을 통해 추가 구성 세부 정보를 제공해야 할 수 있습니다.

10.3. 백엔드 모드 공유

각 공유 드라이버는 사용 가능한 드라이버 모드 중 하나를 지원합니다.

- 공유 서버 - **driver_handles_share_servers = True** - 공유 드라이버는 공유 서버를 생성하고 공유 서버 라이프사이클을 관리합니다.

- 공유 서버가 없음 - **driver_handles_share_servers = False** - 관리자(공유 드라이버가 아닌)는 공유 서버의 존재 여부 대신 네트워크 인터페이스를 사용하여 베어 메탈 스토리지를 관리합니다.

공유 서버 모드 없음 - 이 모드에서는 드라이버가 공유 서버를 설정하지 않으므로 새 네트워크 인터페이스를 설정할 필요가 없습니다. 드라이버에서 관리하는 스토리지 컨트롤러에 필요한 모든 네트워크 인터페이스가 있다고 가정합니다. 드라이버는 이전에 공유 서버를 생성하지 않고 공유를 직접 생성합니다. 이 모드에서 작동하는 드라이버를 사용하여 공유를 생성하기 위해 **manila**는 사용자가 개인 공유 네트워크를 생성할 필요가 없습니다.

참고

공유 서버 모드가 없는 경우 **manila**는 모든 프로젝트에서 공유한 모든 공유에 이미 연결할 수 있는 네트워크 인터페이스에 있다고 가정합니다.

no 공유 서버 모드에서 공유 드라이버는 공유 서버 라이프사이클을 처리하지 않습니다. 관리자는 프로젝트 격리를 제공하는 데 필요할 수 있는 스토리지, 네트워킹 및 기타 호스트 측 구성을 처리해야 합니다. 이 모드에서는 관리자가 공유를 내보내는 호스트로 스토리지를 설정할 수 있습니다. **OpenStack** 클라우드 내의 모든 프로젝트는 공통 네트워크 파이프를 공유합니다. 격리가 부족하면 보안 및 서비스 품질에 영향을 미칠 수 있습니다. 공유 서버를 처리하지 않는 공유 드라이버를 사용하는 경우 클라우드 사용자는 트리에서 신뢰할 수 없는 사용자가 해당 공유에 액세스할 수 없는지 확인할 수 없습니다. 공용 클라우드에서는 한 클라이언트에서 모든 네트워크 대역폭을 사용할 수 있으므로 관리자는 이러한 상황이 발생하지 않도록 주의해야 합니다. 네트워크 분산은 **OpenStack** 툴뿐 아니라 어떠한 방법으로도 수행할 수 있습니다.

공유 서버 모드 - 이 모드에서 드라이버는 공유 서버를 생성하고 기존 **OpenStack** 네트워크에 연결할 수 있습니다. **Manila**는 새 공유 서버가 필요한지 여부를 확인하고, 공유 드라이버가 필수 공유 서버를 생성하는 데 필요한 모든 네트워킹 정보를 제공합니다.

공유 서버를 처리하는 드라이버 모드에서 공유를 생성하는 경우 공유가 내보낼 것으로 예상되는 공유 네트워크를 제공해야 합니다. **Manila**는 이 네트워크를 사용하여 이 네트워크의 공유 서버에 대한 네트워크 포트를 만듭니다.

사용자는 공유 서버 및 공유 서버 백엔드 모드 모두에서 보안 서비스를 구성할 수 있습니다. 그러나 공유 서버가 없는 백엔드 모드에서는 관리자가 호스트에서 필요한 인증 서비스를 수동으로 설정해야 합니다. 공유 서버 모드에서 **manila**는 사용자가 생성한 공유 서버의 보안 서비스를 구성할 수 있습니다.

10.4. MANILA의 네트워킹 요구 사항

Manila는 flat,GRE,VLAN,VXLAN 등 다양한 네트워크 유형과 통합할 수 있습니다.

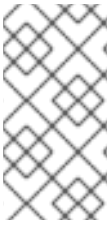
참고

Manila는 네트워크 프로바이더가 제공하는 실제 네트워크와 함께 네트워크 정보를 데이터베이스에만 저장합니다. Manila는 OpenStack Networking 서비스(neutron) 및 "독립형" 사전 구성된 네트워킹도 사용할 수 있도록 지원합니다.

공유 서버 백엔드 모드에서 공유 드라이버는 각 공유 네트워크에 대한 공유 서버를 생성하고 관리합니다. 이 모드는 두 가지 변형으로 나눌 수 있습니다.

- 공유 서버의 백엔드 모드의 플랫 네트워크
- 공유 서버의 백엔드 모드에서 분할된 네트워크

사용자는 OpenStack Networking(neutron) 서비스의 네트워크 및 서브넷을 사용하여 공유 네트워크를 만들 수 있습니다. StandAloneNetworkPlugin 을 사용하기로 결정한 경우 관리자가 구성 파일에서 이를 사전 구성하므로 사용자는 네트워킹 정보를 제공하지 않아도 됩니다.



참고

일부 공유 드라이버에서 생성한 공유 서버는 계산 서비스로 생성된 계산 서버입니다. 이러한 드라이버 중 일부는 네트워크 플러그인을 지원하지 않습니다.

공유 네트워크가 생성되면 manila는 네트워크 프로바이더가 결정한 네트워크 정보(네트워크 유형, 세그먼트 식별자(네트워크에서 분할을 사용하는 경우) 및 네트워크를 할당할 CIDR 표기법의 IP 블록을 검색합니다.

사용자는 AD 또는 LDAP 도메인 또는 Kerberos 영역과 같은 보안 요구 사항을 지정하는 보안 서비스를 생성할 수 있습니다. Manila에서는 보안 서비스에서 참조하는 모든 호스트에 공유 서버가 생성된 서브넷에서 연결할 수 있다고 가정하여 이 모드를 사용할 수 있는 사례 수를 제한합니다.

참고

일부 공유 드라이버는 모든 유형의 분할을 지원하지 않을 수 있습니다. 자세한 내용은 사용 중인 드라이버의 사양을 참조하십시오.

10.5. MANILA를 사용한 보안 서비스

Manila는 네트워크 인증 프로토콜과 통합되어 파일 공유에 대한 액세스를 제한할 수 있습니다. 각 프로젝트에는 클라우드의 keystone 인증 도메인과 별도로 작동하는 자체 인증 도메인이 있을 수 있습니다. 이 프로젝트 도메인을 사용하여 manila를 포함하여 OpenStack 클라우드 내에서 실행되는 애플리케이션에 권한 부여(AuthZ) 서비스를 제공할 수 있습니다. 사용 가능한 인증 프로토콜에는 LDAP, Kerberos, Microsoft Active Directory 인증 서비스가 포함됩니다.

10.5.1. 보안 서비스 소개

공유를 만들고 내보내기 위치를 가져온 후에는 사용자가 마운트하고 파일을 사용할 수 있는 권한이 없습니다. 사용자는 새 공유에 대한 액세스 권한을 명시적으로 부여해야 합니다.

클라이언트 인증 및 권한 부여(authN/authZ)는 보안 서비스와 함께 수행할 수 있습니다. Manila는 공유 드라이버 및 백엔드에서 지원하는 경우 LDAP, Kerberos 또는 Microsoft Active 디렉터리를 사용할 수 있습니다.

참고

경우에 따라 보안 서비스 중 하나를 명시적으로 지정해야 합니다. 예를 들어 NetApp, EMC 및 Windows 드라이버는 CIFS 프로토콜을 사용하여 공유를 생성하기 위해 Active Directory가 필요합니다.

10.5.2. 보안 서비스 관리

*보안 서비스*는 Active Directory 도메인 또는 Kerberos 도메인과 같은 특정 공유 파일 시스템 프로토콜에 대한 보안 영역을 정의하는 옵션 집합을 추상화하는 manila 엔터티입니다. 보안 서비스에는 지정된 도메인을 연결하는 서버를 생성하는 데 필요한 모든 정보가 포함되어 있습니다.

사용자는 API를 사용하여 보안 서비스를 생성, 업데이트, 보기 및 삭제할 수 있습니다. 보안 서비스는 다음과 같은 가정을 기반으로 설계되었습니다.

- 프로젝트는 보안 서비스에 대한 세부 정보를 제공합니다.

- 관리자는 보안 서비스에 중점을 두고 이러한 보안 서비스의 서버 측면을 구성합니다.
- manila API 내부에서 security_service 는 share_networks 와 연결됩니다.
- 공유 드라이버는 보안 서비스의 데이터를 사용하여 새로 만든 공유 서버를 구성합니다.

보안 서비스를 생성할 때 다음 인증 서비스 중 하나를 선택할 수 있습니다.

- LDAP - 경량 디렉터리 액세스 프로토콜. IP 네트워크를 통해 분산 디렉터리 정보 서비스에 액세스하고 유지 관리하는 애플리케이션 프로토콜입니다.
- Kerberos - 안전하지 않은 네트워크를 통해 통신하는 노드가 안전한 방식으로 서로의 ID를 증명할 수 있도록 티켓에 따라 작동하는 네트워크 인증 프로토콜입니다.
- Active Directory - Microsoft가 Windows 도메인 네트워크용으로 개발한 디렉터리 서비스입니다. 는 LDAP, Microsoft의 Kerberos 버전 및 DNS를 사용합니다.

Manila를 사용하면 다음 옵션을 사용하여 보안 서비스를 구성할 수 있습니다.

- 프로젝트 네트워크 내에서 사용되는 DNS IP 주소입니다.
- 보안 서비스의 IP 주소 또는 호스트 이름.
- 보안 서비스의 도메인입니다.
- 프로젝트에서 사용하는 사용자 또는 그룹 이름입니다.
- 사용자 이름을 지정하는 경우 사용자의 암호입니다.

기존 보안 서비스 엔터티는 manila에 공유 그룹의 보안 및 네트워크 구성에 대해 알리는 공유 네트워

크 엔터티와 연결할 수 있습니다. 지정된 공유 네트워크의 모든 보안 서비스 목록을 보고 공유 네트워크에서 연결을 끊을 수도 있습니다.

공유 소유자인 관리자 및 사용자는 **IP** 주소, 사용자, 그룹 또는 **TLS** 인증서를 통해 인증으로 액세스 규칙을 생성하여 공유에 대한 액세스를 관리할 수 있습니다. 인증 방법은 구성 및 사용 중인 드라이버 및 보안 서비스를 공유하는 데 따라 다릅니다. 그런 다음 **manila** 및 **keystone** 없이 클라이언트와 작동할 수 있는 특정 인증 서비스를 사용하도록 백엔드를 구성할 수 있습니다.

참고

다양한 공유 드라이버에서 다양한 인증 서비스를 지원합니다. 다른 드라이버별 기능 지원에 대한 자세한 내용은 https://docs.openstack.org/manila/latest/admin/share_back_ends_feature_support_mapping.html를 참조하십시오.

드라이버의 특정 인증 서비스에 대한 지원은 공유 파일 시스템 프로토콜로 구성할 수 있음을 의미하지는 않습니다. 지원되는 공유 파일 시스템 프로토콜은 **NFS, CEPHFS, CIFS, GlusterFS, HDFS**입니다. 특정 드라이버 및 보안 서비스 구성에 대한 자세한 내용은 드라이버 벤더의 설명서를 참조하십시오.

일부 드라이버는 보안 서비스를 지원하며 기타 드라이버는 위에서 언급한 보안 서비스를 지원하지 않습니다. 예를 들어 **NFS** 또는 **CIFS** 공유 파일 시스템 프로토콜이 있는 일반 드라이버는 **IP** 주소를 통한 인증 방법만 지원합니다.

참고

대부분의 경우 **CIFS** 공유 파일 시스템 프로토콜을 지원하는 드라이버는 **Active Directory**를 사용하고 사용자 인증을 통해 액세스를 관리하도록 구성할 수 있습니다.

- **GlusterFS** 프로토콜을 지원하는 드라이버는 **TLS** 인증서를 사용하여 인증과 함께 사용할 수 있습니다.
- **IP** 주소를 사용하여 **NFS** 프로토콜 인증을 지원하는 드라이버가 유일한 지원되는 옵션입니다.
- **HDFS** 공유 파일 시스템 프로토콜은 **NFS** 액세스를 사용하므로 **IP** 주소를 사용하여 인증하도록 구성할 수도 있습니다.

프로덕션 **manila** 배포에 권장되는 구성은 **CIFS** 공유 프로토콜로 공유를 생성하고 **Microsoft Active Directory** 디렉터리 서비스에 추가하는 것입니다. 이 구성을 사용하면 중앙 집중식 데이터베이스와 **Kerberos** 접근 방식을 통합하는 서비스를 받게 됩니다.

10.6. 공유 액세스 제어

사용자는 생성한 공유에 대한 액세스 권한이 있는 특정 클라이언트를 지정할 수 있습니다. **keystone** 서비스로 인해 개별 사용자가 생성한 공유는 동일한 프로젝트 내의 자체 및 기타 사용자에게만 표시됩니다. **Manila**를 사용하면 사용자가 표시되는 "공개로" 공유를 만들 수 있습니다. 이러한 공유는 소유자가 액세스 권한을 부여하는 경우 다른 **OpenStack** 프로젝트에 속하는 사용자의 대시보드에 표시됩니다. 네트워크에서 액세스할 수 있게 되면 이러한 공유를 마운트할 수도 있습니다.

공유를 생성하는 동안, **--public** 키를 사용하여 다른 프로젝트에서 공유 목록에서 이를 확인하고 자세한 정보를 볼 수 있도록 공개합니다.

policy.json 파일에 따라, 관리자 및 공유 소유자인 사용자는 액세스 규칙을 생성하는 방법으로 공유에 대한 액세스를 관리할 수 있습니다. **manila access-allow**,**manila access-deny** 및 **manila access-list** 명령을 사용하여 지정된 공유에 대한 액세스 권한을 부여, 거부 및 나열할 수 있습니다.

참고

Manila는 스토리지 시스템의 포괄적인 관리를 제공하지 않습니다. 백엔드 시스템을 권한 없는 액세스로부터 별도로 보호해야 합니다. 결과적으로 백엔드 스토리지 장치를 손상시켜 대역 액세스에서 얻을 수 있는 경우 **manila API**에서 제공하는 보호 조치를 피할 수 있습니다.

공유가 방금 생성되면 연결된 기본 액세스 규칙과 마운트 권한이 없습니다. 이는 내보내기 프로토콜이 사용 중인 구성을 마운트할 때 볼 수 있습니다. 예를 들어 스토리지에는 각 원격 공유를 제어하고 액세스할 수 있는 호스트를 정의하는 **NFS** 명령 **exportfs** 또는 **/etc/exports** 파일이 있습니다. 공유를 마운트할 수 없는 경우 비어 있습니다. 원격 **CIFS** 서버의 경우 설정을 표시하는 **net conf list** 명령이 있습니다. **hosts deny** 매개 변수는 공유 드라이버에서 **0.0.0.0/0** 으로 설정해야 합니다. 즉, 공유를 마운트하기 위해 모든 호스트가 거부됩니다.

manila를 사용하면 지원되는 공유 액세스 수준 중 하나를 지정하여 공유에 대한 액세스 권한을 부여하거나 거부할 수 있습니다.

- **RW - RW**(읽기 및 쓰기) 액세스. 이는 기본값입니다.

- **ro-** 읽기 전용(RO) 액세스.

참고

RO 액세스 수준은 관리자가 일부 특정 편집기 또는 기여자에게 RW(읽기 및 쓰기) 액세스 권한을 제공하고 나머지 사용자(뷰어)에 대해 읽기 전용(RO) 액세스 권한을 부여하는 경우 공용 공유에서 유용할 수 있습니다.

또한 지원되는 인증 방법 중 하나를 지정해야 합니다.

- **IP - IP** 주소를 사용하여 인스턴스를 인증합니다. IP 액세스는 CIDR 표기법으로 표시되는 잘 구성된 IPv4 또는 IPv6 주소 또는 서브넷으로 주소 지정 가능한 클라이언트에 제공할 수 있습니다.

- 인증서 **- TLS** 인증서를 사용하여 인스턴스를 인증합니다. TLS ID를 IDENTKEY 로 지정합니다. 유효한 값은 인증서의 CN(일반 이름)에서 최대 64자까지 문자열입니다.

- **user -** 지정된 사용자 또는 그룹 이름으로 인증합니다. 유효한 값은 일부 특수 문자를 포함할 수 있으며 4~32자 길이의 영숫자 문자열입니다.

참고

지원되는 인증 방법은 사용하는 드라이버, 보안 서비스 및 공유 파일 시스템 프로토콜을 공유하는 데 따라 다릅니다. 지원되는 공유 파일 시스템 프로토콜은 MapRFS, CEPHFS, NFS, CIFS, GlusterFS, HDFS입니다. 지원되는 보안 서비스는 LDAP, Kerberos 프로토콜 또는 Microsoft Active Directory 서비스입니다.

공유에 대한 액세스 규칙(ACL)이 올바르게 구성되었는지 확인하려면 해당 권한을 나열할 수 있습니다.

참고

공유에 대한 보안 서비스를 선택할 때 공유 드라이버가 사용 가능한 인증 방법을 사용하여 액세스 규칙을 만들 수 있는지 확인해야 합니다. 지원되는 보안 서비스는 LDAP, Kerberos, Microsoft Active Directory입니다.

10.7. 공유 유형 액세스 제어

공유 유형은 프로젝트에 표시된 설명으로 구성된 관리자 정의 *서비스* 유형과 *추가 사양* 이라는 프로젝트 가시성이 없는 키-값 쌍 목록입니다. **manila-scheduler** 는 추가 사양을 사용하여 스케줄링 결정을 내리고 드라이버는 공유 생성을 제어합니다.

관리자는 공유 유형을 생성 및 삭제할 수 있으며 **manila** 내부에서 의미를 부여하는 추가 사양을 관리할 수도 있습니다. 프로젝트는 공유 유형을 나열할 수 있으며 이를 사용하여 새 공유를 만들 수 있습니다. 공유 유형은 공용 및 개인용으로 만들 수 있습니다. 이는 다른 프로젝트에서 공유 유형 목록에서 볼 수 있는지 여부를 정의하는 공유 유형에 대한 가시성 수준이며, 이를 사용하여 새 공유를 만듭니다.

기본적으로 공유 유형은 공용으로 생성됩니다. 공유 유형을 생성하는 동안 **--is_public** 매개 변수를 **False** 로 설정하여 공유 유형을 비공개로 설정하여 다른 프로젝트가 공유 유형 목록에서 보고 새 공유를 생성하지 못하도록 합니다. 반면, 공용 공유 유형은 클라우드의 모든 프로젝트에서 사용할 수 있습니다.

Manila를 사용하면 관리자가 프로젝트의 개인 공유 유형에 대한 액세스 권한을 부여하거나 거부할 수 있습니다. 지정된 개인 공유 유형의 액세스에 대한 정보를 가져올 수도 있습니다.

참고

추가 사양으로 인한 공유 유형은 사용자가 공유를 생성하기 전에 백엔드를 필터링하거나 선택하는 데 도움이 되기 때문에 공유 유형에 대한 액세스를 사용하여 특정 백엔드를 선택할 수 있습니다.

예를 들어 **admin** 프로젝트의 관리자는 **my_type** 이라는 개인 공유 유형을 생성하여 목록에 볼 수 있습니다. 아래 콘솔 예제에서 로그인 및 로그아웃이 생략되고 현재 로그인한 사용자를 표시하도록 환경 변수가 제공됩니다.

```
$ env | grep OS_
...
OS_USERNAME=admin
OS_TENANT_NAME=admin
...
$ manila type-list --all
+----+--------+-----------+-----------+---------------------------------+----------------------+
| ID | Name   | Visibility| is_default| required_extra_specs            | optional_extra_specs |
+----+--------+-----------+-----------+---------------------------------+----------------------+
| 4..| my_type| private   | -         | driver_handles_share_servers:False| snapshot_support:True |
| 5..| default| public    | YES       | driver_handles_share_servers:True | snapshot_support:True |
+----+--------+-----------+-----------+---------------------------------+----------------------+
```

demo 프로젝트의 **demo** 사용자는 유형을 나열할 수 있으며 **my_type** 이라는 개인 공유 유형이 표시되지 않습니다.

```
$ env | grep OS_
...
OS_USERNAME=demo
OS_TENANT_NAME=demo
...
$ manila type-list --all
+----+--------+-----------+-----------+---------------------------------+----------------------+
| ID | Name   | Visibility| is_default| required_extra_specs            | optional_extra_specs |
+----+--------+-----------+-----------+---------------------------------+----------------------+
| 5..| default| public    | YES       | driver_handles_share_servers:True| snapshot_support:True|
+----+--------+-----------+-----------+---------------------------------+----------------------+
```

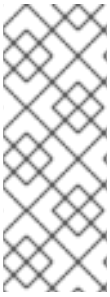
관리자는 **df29a37db5ae48d19b349fe947fada46** 과 동일한 프로젝트 **ID**를 사용하여 **demo** 프로젝트의 개인 공유 유형에 대한 액세스 권한을 부여할 수 있습니다.

```
$ env | grep OS_
...
OS_USERNAME=admin
OS_TENANT_NAME=admin
...
$ openstack project list
+----------------------------------+--------------------+
| ID                               | Name               |
+----------------------------------+--------------------+
| ...                              | ...                |
| df29a37db5ae48d19b349fe947fada46 | demo               |
+----------------------------------+--------------------+
$ manila type-access-add my_type df29a37db5ae48d19b349fe947fada46
```

결과적으로 **demo** 프로젝트의 사용자는 **private** 공유 유형을 보고 공유 생성에 사용할 수 있습니다.

```
$ env | grep OS_
...
OS_USERNAME=demo
OS_TENANT_NAME=demo
...
$ manila type-list --all
+----+--------+-----------+-----------+---------------------------------+----------------------+
| ID | Name   | Visibility| is_default| required_extra_specs            | optional_extra_specs  |
+----+--------+-----------+-----------+---------------------------------+----------------------+
| 4..| my_type| private   | -         | driver_handles_share_servers:False| snapshot_support:True |
| 5..| default| public    | YES       | driver_handles_share_servers:True | snapshot_support:True |
+----+--------+-----------+-----------+---------------------------------+----------------------+
```

지정된 프로젝트의 액세스를 거부하려면 **manila type-access-remove <share_type> <project_id>** 를 사용합니다.



참고

공유 유형의 목적을 보여주는 예를 들어 두 개의 백엔드가 있는 상황을 고려하십시오. 공용 스토리지로서의 **LVM,** 프라이빗 스토리지로서의 **Ceph.** 이 경우 특정 프로젝트에 대한 액세스 권한을 부여하고 사용자/그룹 인증 방법을 사용하여 액세스를 제어할 수 있습니다.

10.8. POLICIES

공유 파일 시스템 서비스 **API**는 역할 기반 액세스 제어 정책으로 제어됩니다. 이러한 정책은 특정 방식으로 특정 **API**에 액세스할 수 있으며 서비스의 **policy.json** 파일에 정의된 사용자를 결정합니다.

참고

구성 파일 **policy.json** 은 어디서나 배치할 수 있습니다. 경로 **/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 은 기본적으로 예상됩니다.

manila에 대한 **API** 호출이 발생할 때마다 정책 엔진은 적절한 정책 정의를 사용하여 호출을 수락할 수 있는지 확인합니다. 정책 규칙은 **API** 호출이 허용되는 상황을 결정합니다. **/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 파일에는 규칙이 빈 문자열인 경우 **"";** 사용자 역할 또는 규칙을 기반으로 하는 규칙, 부울 표현식이 포함된 규칙이 항상 허용되는 규칙이 있습니다. 다음은 **manila**에 대한 **policy.json** 파일의 조각입니다. **OpenStack** 릴리스 간에 변경될 수 있습니다.

```
{
    "context_is_admin": "role:admin",
    "admin_or_owner": "is_admin:True or project_id:%(project_id)s",
    "default": "rule:admin_or_owner",
    "share_extension:quotas:show": "",
    "share_extension:quotas:update": "rule:admin_api",
    "share_extension:quotas:delete": "rule:admin_api",
    "share_extension:quota_classes": "",
}
```

사용자는 정책에서 참조하는 그룹 및 역할에 할당되어야 합니다. 이 작업은 사용자 관리 명령을 사용할 때 서비스에 의해 자동으로 수행됩니다.

참고

/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json 에 대한 변경 사항은 즉시 유효하므로 **manila**가 실행되는 동안 새 정책을 구현할 수 있습니다. 정책을 수동으로 수정하면 예기치 않은 부작용이 발생할 수 있으며 권장되지 않습니다. **Manila**는 기본 정책 파일을 제공하지 않습니다. 모든 기본 정책은 코드 베이스 내에 있습니다. 다음을 실행하여 **manila** 코드에서 기본 정책을 생성할 수 있습니다. **oslopolicy-sample-generator --config-file=var/lib/config-data/puppet-generated/manila/etc/manila/manila-policy-generator.conf**

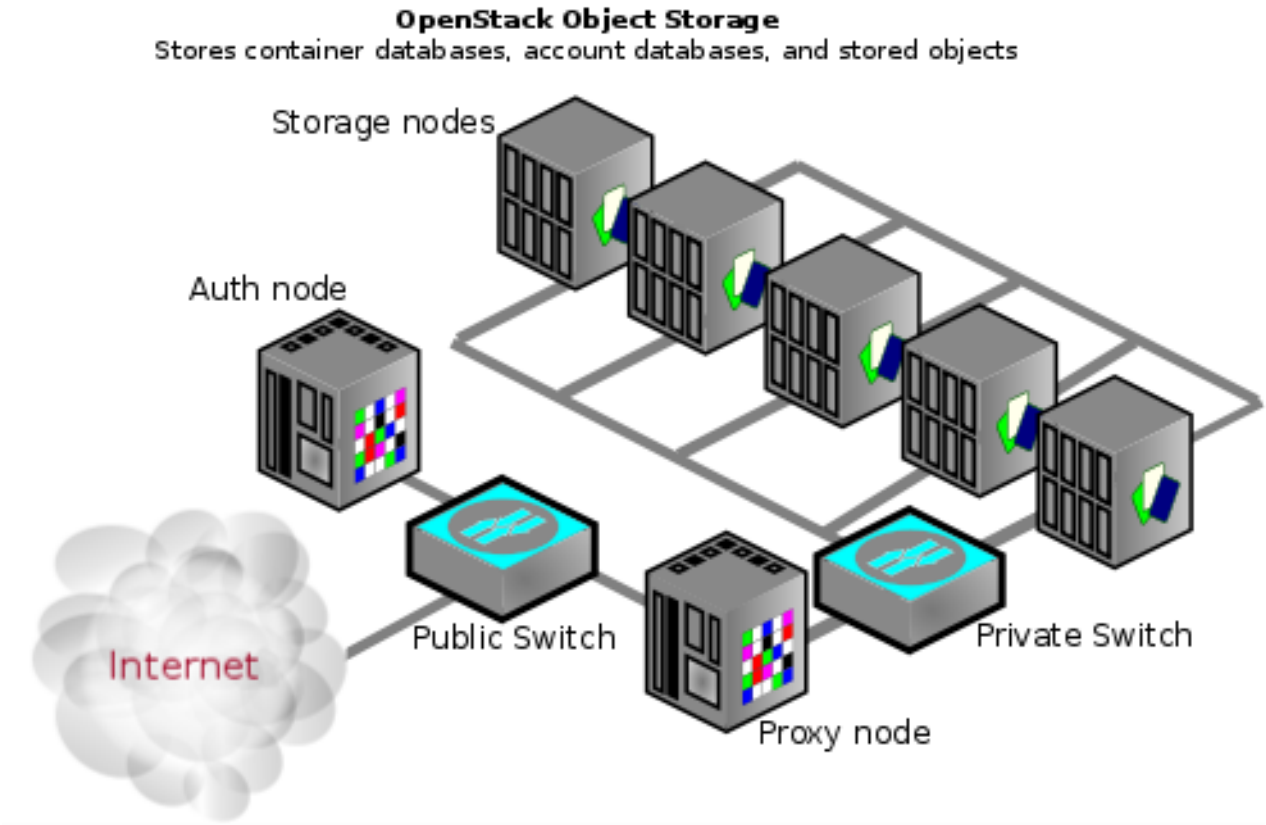
11장. 오브젝트 스토리지

Object Storage(swift) 서비스는 **HTTP**를 통해 데이터를 저장하고 검색합니다. 개체(브로브 데이터)는 익명의 읽기 전용 액세스, **ACL** 정의 액세스 또는 임시 액세스를 제공하도록 구성할 수 있는 조직 계층 구조에 저장됩니다. **Swift**는 미들웨어를 통해 구현된 여러 토큰 기반 인증 메커니즘을 지원합니다.

애플리케이션은 업계 표준 **HTTP RESTful API**를 사용하여 오브젝트 스토리지에 데이터를 저장하고 검색합니다. 백엔드 **swift** 구성 요소는 동일한 **RESTful** 모델을 따릅니다. 일부 **API**(예: 지속성 관리)는 클러스터에 비공개로 유지됩니다.

swift의 구성 요소는 다음과 같은 기본 그룹으로 나뉩니다.

- 프록시 서비스
- 인증 서비스
- 스토리지 서비스
  - 계정 서비스
  - 컨테이너 서비스
  - 오브젝트 서비스



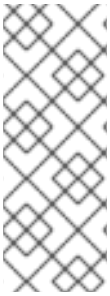
참고

오브젝트 스토리지 설치는 인터넷에 연결할 필요가 없으며 조직의 내부 네트워크 인프라의 일부인 공용 스위치가 있는 프라이빗 클라우드일 수도 있습니다.

11.1. 네트워크 보안

swift의 보안 강화는 네트워킹 구성 요소 보안과 함께 시작됩니다. 자세한 내용은 네트워킹 장을 참조하십시오.

고가용성을 위해 **rsync** 프로토콜을 사용하여 스토리지 서비스 노드 간에 데이터를 복제합니다. 또한 프록시 서비스는 클라이언트 엔드포인트와 클라우드 환경 간에 데이터를 릴레이할 때 스토리지 서비스와 통신합니다.



참고

Swift는 노드 간 통신에서 암호화 또는 인증을 사용하지 않습니다. **swift**는 성능상의 이유로 기본 **rsync** 프로토콜을 사용하고 **rsync** 통신에 **SSH**를 사용하지 않기 때문입니다. 따라서 아키텍처 다이어그램에 프라이빗 스위치 또는 사설 네트워크(**[V]LAN**)가 표시됩니다. 이 데이터 영역은 다른 **OpenStack** 데이터 네트워크와도 분리되어야 합니다.

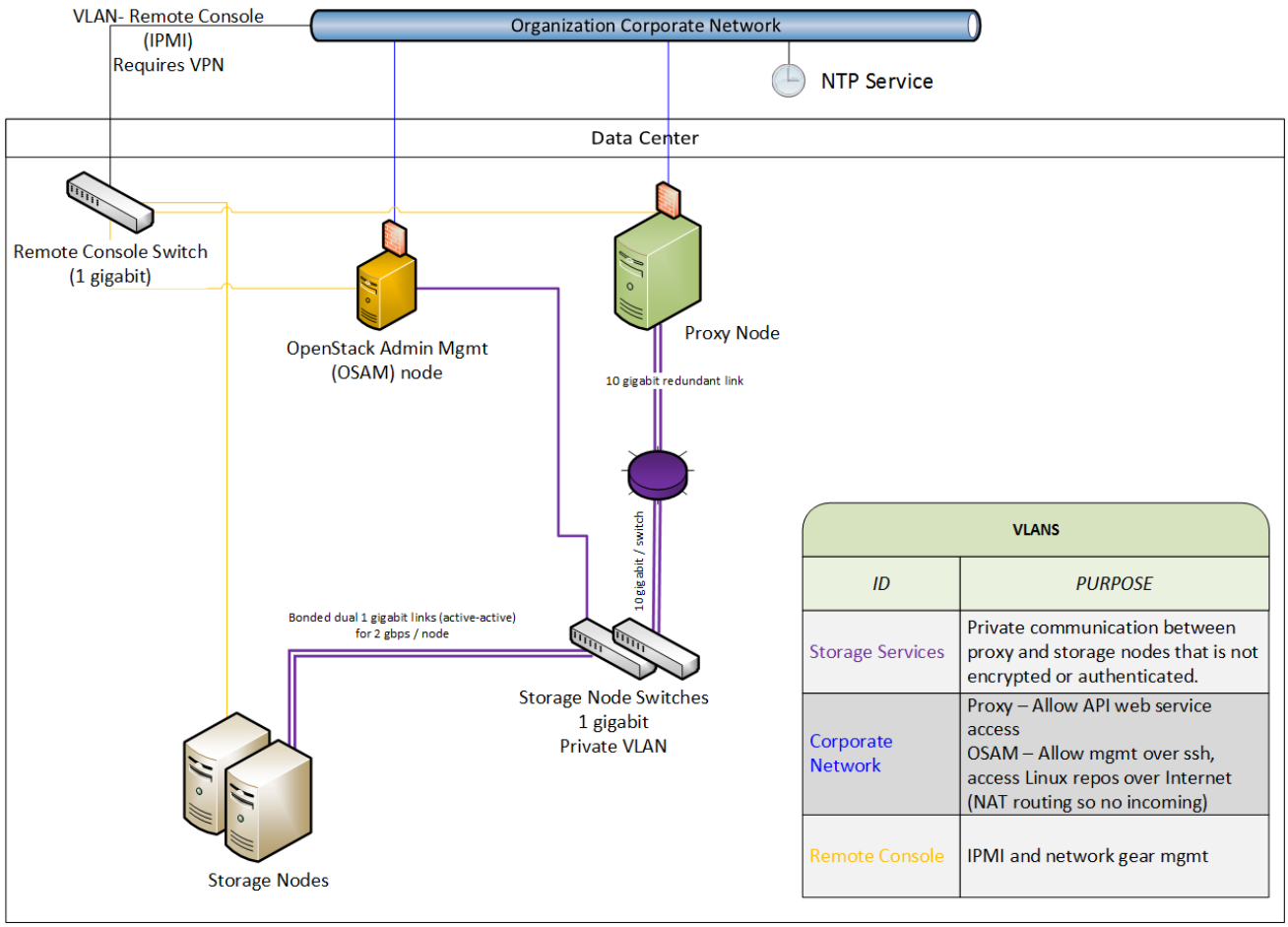
참고

데이터 영역의 스토리지 노드에 프라이빗(V)LAN 네트워크 세그먼트를 사용합니다.

이를 위해서는 프록시 노드에 듀얼 인터페이스(물리적 또는 가상)가 있어야 합니다.

- 소비자가 연결할 수 있는 공용 인터페이스로서의 인터페이스 한 개입니다.
- 스토리지 노드에 액세스할 수 있는 사설 인터페이스로서의 또 다른 인터페이스.

다음 그림은 관리 노드(OSAM)와 함께 Object Storage 네트워크 아키텍처를 사용하여 가능한 하나의 네트워크 아키텍처를 보여줍니다.



11.2. 일반 서비스 보안

11.2.1. 루트가 아닌 사용자로 서비스 실행

루트가 아닌(uid0) 서비스 계정에서 실행되도록 **swift**를 구성하는 것이 좋습니다. 한 가지 권장 사항은 **director** 에서 배포한 대로 기본 그룹 **swift** 의 사용자 이름 **swift**입니다. 오브젝트 스토리지 서비스에는 **proxy-server, container -server,account-server** 와 같은 서비스가 포함됩니다.

11.2.2. 파일 권한

/var/lib/config-data/puppet-generated/swift/etc/swift/ 디렉터리에는 링 토폴로지 및 환경 구성에 대한 정보가 포함되어 있습니다. 권장되는 권한은 다음과 같습니다.

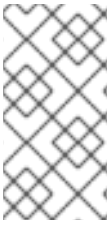
```
chown -R root:swift /var/lib/config-data/puppet-generated/swift/etc/swift/*
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type f -exec chmod 640 {} \;
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type d -exec chmod 750 {} \;
```

이 제한으로 인해 **root**는 **swift** 그룹의 멤버십으로 인해 서비스가 계속 읽을 수 있는 구성 파일을 수정할 수 있습니다.

11.3. 스토리지 서비스 보안

다음은 다양한 스토리지 서비스에 대한 기본 수신 대기 포트입니다.

- 계정 서비스 **- TCP/6002**
- 컨테이너 서비스 **- TCP/6001**
- 개체 서비스 **- TCP/**
- **Rsync - TCP/873**



참고

rsync 대신 **ssync**를 사용하면 지속성을 유지하는 데 오브젝트 서비스 포트가 사용됩니다.

참고

스토리지 노드에서는 인증이 발생하지 않습니다. 이러한 포트 중 하나에서 스토리지 노드에 연결할 수 있는 경우 인증 없이 데이터에 액세스하거나 수정할 수 있습니다. 이 문제를 완화하려면 이전에 사설 스토리지 네트워크 사용에 대한 권장 사항을 따라야 합니다.

11.3.1. Object Storage 계정 용어

swift 계정은 사용자 계정 또는 자격 증명이 아닙니다. 다음과 같은 차이점이 있습니다.

- Swift 계정 - 컨테이너 컬렉션(사용자 계정 또는 인증이 아님). 사용하는 인증 시스템은 계정과 연결된 사용자 및 액세스 방법을 결정합니다.
- Swift 컨테이너 - 오브젝트 컬렉션입니다. 컨테이너의 메타데이터는 ACL에 사용할 수 있습니다. ACL은 사용된 인증 시스템에 따라 다릅니다.
- Swift 개체 - 실제 데이터 개체입니다. 개체 수준의 ACL은 메타데이터에서도 사용할 수 있으며 사용되는 인증 시스템에 따라 다릅니다.

각 수준에서는 사용자 액세스를 제어하는 ACL이 있습니다. 사용 중인 인증 시스템에 따라 ACL이 해석됩니다. 가장 일반적인 인증 프로바이더 유형은 ID 서비스(keystone)입니다. 사용자 지정 인증 공급자도 사용할 수 있습니다.

11.4. 프록시 서비스 보안

프록시 노드에는 두 개 이상의 인터페이스(물리적 또는 가상)(공용 및 개인용)가 있어야 합니다. 방화벽 또는 서비스 바인딩을 사용하여 공용 인터페이스를 보호할 수 있습니다. 공용 서비스는 엔드포인트 클라이언트 요청을 처리하고, 인증하고, 적절한 작업을 수행하는 HTTP 웹 서버입니다. 사설 인터페이스에는 수신 대기 서비스가 필요하지 않지만, 사설 스토리지 네트워크의 스토리지 노드에 대한 발신 연결을 설정하는 데 사용됩니다.

11.4.1. HTTP 수신 포트

director는 루트가 아닌 사용자(UID 0) 사용자로 실행되도록 웹 서비스를 구성합니다. 1024 보다 큰 포트 번호를 사용하면 웹 컨테이너의 모든 부분을 root로 실행할 수 없습니다. 일반적으로 HTTP REST API를 사용하고 자동 인증을 수행하는 클라이언트는 인증 응답에서 필요한 전체 REST API URL을 검색합니다. OpenStack REST API를 사용하면 클라이언트가 하나의 URL로 인증한 다음 실제 서비스에 완전히 다른 URL을 사용하도록 리디렉션됩니다. 예를 들어 클라이언트는 https://identity.cloud.example.org:55443/v1/auth 에 인증하고 인증 키 및 스토리지 URL (프록시 노드

또는 로드 밸런서의 URL)을 사용하여 응답을 받을 수 있습니다.
https://swift.cloud.example.org:44443/v1/AUTH_8980.

11.4.2. 로드 밸런서

Apache 사용 옵션을 사용할 수 없거나 TLS 작업을 오프로드하려는 성능을 위해 전용 네트워크 장치 로드 밸런서를 사용할 수 있습니다. 이는 여러 프록시 노드를 사용할 때 중복성과 로드 밸런싱을 제공하는 일반적인 방법입니다.

TLS를 오프로드하도록 선택하는 경우 네트워크 연결이 로드 밸런서와 프록시 노드 간에 연결되는지 확인하여 네트워크의 다른 노드(sniff)가 암호화되지 않은 트래픽을 중단할 수 없도록 개인(V)LAN 세그먼트에 있는지 확인합니다. 이러한 위반이 발생하는 경우 공격자는 엔드포인트 클라이언트 또는 클라우드 관리자 자격 증명에 액세스하고 클라우드 데이터에 액세스할 수 있습니다.

사용하는 인증 서비스는 끝점 클라이언트에 대한 응답에서 다른 URL을 구성하는 방법을 결정하여 개별 프록시 노드 대신 로드 밸런서를 사용할 수 있습니다.

11.5. 오브젝트 스토리지 인증

Object Storage(swift)는 WSGI 모델을 사용하여 일반 확장성을 제공할 뿐만 아니라 엔드포인트 클라이언트의 인증에도 사용되는 미들웨어 기능을 제공합니다. 인증 프로바이더는 어떤 역할 및 사용자 유형이 존재하는지 정의합니다. 일부는 기존 사용자 이름 및 암호 자격 증명을 사용하는 반면, 다른 일부는 API 키 토큰 또는 클라이언트측 x.509 인증서를 활용할 수 있습니다. 사용자 지정 공급업체는 사용자 지정 미들웨어를 사용하여 통합할 수 있습니다.

오브젝트 스토리지에는 기본적으로 두 개의 인증 미들웨어 모듈이 제공되며, 둘 중 하나를 사용자 지정 인증 미들웨어 개발을 위한 샘플 코드로 사용할 수 있습니다.

11.5.1. Keystone

Keystone은 OpenStack에서 일반적으로 사용되는 ID 프로바이더입니다. 개체 스토리지의 인증에도 사용할 수 있습니다.

11.6. 유휴 시 SWIFT 오브젝트 암호화

Swift는 Barbican과 통합하여 저장된(수신) 오브젝트를 투명하게 암호화하고 암호를 해독할 수 있습니다. 대기 중인 암호화는 전송 중인 암호화와 별개이며 디스크에 저장되는 동안 암호화되는 오브젝트를 나타냅니다.

Swift는 swift에 업로드할 때 오브젝트가 자동으로 암호화되는 상태에서 이러한 암호화 작업을 투명하게 수행한 다음 사용자에게 제공할 때 자동으로 암호 해독됩니다. 이 암호화 및 암호 해독은 Barbican에 저장되는 동일한 (대칭) 키를 사용하여 수행됩니다.

자세한 내용은 Barbican 통합 가이드를 참조하십시오.
https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html-single/manage_secrets_with_openstack_key_manager/

11.7. 추가 항목

모든 노드의 /var/lib/config-data/puppet-generated/swift/etc/swift/swift.conf 에는 swift_hash_path_prefix 설정과 swift_hash_path_suffix 설정이 있습니다. 이러한 기능이 제공되어 오브젝트의 해시 충돌 가능성을 줄이고 한 사용자가 다른 사용자의 데이터를 덮어씁니다.

이 값은 암호화 방식으로 안전한 난수 생성기를 사용하여 처음에 설정하고 모든 노드에서 일관성 있게 설정해야 합니다. 적절한 ACL로 보호되고 데이터 손실을 방지하기 위해 백업 사본이 있는지 확인합니다.

12장. 모니터링 및 로깅

로그 관리는 OpenStack 배포의 보안 상태를 모니터링하는 데 중요한 구성 요소입니다. 로그는 OpenStack 배포를 구성하는 구성 요소 활동 외에도 관리자, 프로젝트 및 인스턴스의 BAU 조치에 대한 통찰력을 제공합니다.

로그는 사전 예방적 보안과 지속적인 규정 준수 활동에 중요할 뿐만 아니라 조사 및 사고 대응에 있어 중요한 정보 소스이기도 합니다. 예를 들어 keystone 액세스 로그를 분석하면 실패한 로그인, 빈도, 원본 IP, 이벤트가 다른 관련 정보 중에서 계정을 선택하도록 제한되는지 여부를 알릴 수 있습니다.

director에는 AIDE를 사용한 침입 감지 기능과 keystone에 대한 CADF 감사가 포함되어 있습니다. 자세한 내용은 director 강화 장을 참조하십시오.

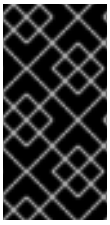
12.1. 모니터링 인프라 강화

중앙 집중식 로깅 시스템은 침입자에게 높은 가치의 대상입니다. 위반이 성공하면 이벤트 기록으로 지우거나 변조할 수 있기 때문입니다. 이를 염두에 두고 모니터링 플랫폼을 강화하는 것이 좋습니다. 또한 가동 중단 또는 DoS가 발생할 경우 페일오버 계획을 통해 이러한 시스템을 정기적으로 백업하는 것이 좋습니다.

12.2. 모니터링 이벤트의 예

이벤트 모니터링은 환경 보안을 위해 보다 사전 예방적인 접근 방식으로 실시간 탐지 및 응답을 제공합니다. 모니터링에 도움이 되는 여러 도구가 있습니다. OpenStack 배포를 위해 하드웨어, OpenStack 서비스 및 클라우드 리소스 사용량을 모니터링해야 합니다.

이 섹션에서는 알아야 할 몇 가지 예제 이벤트에 대해 설명합니다.



중요

이 목록은 전체 목록이 아닙니다. 특정 네트워크에 적용할 수 있고 이종 동작을 고려할 수 있는 추가 사용 사례를 고려해야 합니다.

- 로그 생성이 없음을 감지하는 것은 높은 값 이벤트입니다. 이러한 격차는 서비스 오류 또는 일시적으로 로깅을 끄거나 로그 수준을 수정하여 추적을 숨기는 침입자를 나타낼 수 있습니다.
- 예약되지 않은 시작 또는 중지 이벤트와 같은 애플리케이션 이벤트는 보안에 영향을 미칠 수

있습니다.

- 사용자 로그인 또는 재시작과 같은 **OpenStack** 노드의 운영 체제 이벤트입니다. 따라서 시스템의 적절하고 부적절한 사용을 구분하는 데 있어 중요한 통찰력을 얻을 수 있습니다.

- 네트워크 브리지가 줄어듭니다. 이는 서비스 중단의 위험으로 인해 실행 가능한 이벤트입니다.

- **Compute** 노드에서 **iptables** 플러시 이벤트 및 인스턴스에 대한 액세스가 손실됩니다.

ID 서비스의 사용자, 프로젝트 또는 도메인 삭제에 있는 인스턴스의 보안 위험을 줄이기 위해 시스템에서 알림을 생성하기 위한 논의가 있으며, **OpenStack** 구성 요소가 인스턴스 종료, 연결된 볼륨 연결 해제, **CPU** 및 스토리지 리소스 회수 등의 적절한 이벤트와 같이 이러한 이벤트에 응답하도록 합니다.

침입 감지 소프트웨어, 바이러스 소프트웨어, **computerware** 탐지 및 제거 유틸리티와 같은 보안 모니터링 제어는 공격 또는 침입 발생 시기 및 방법을 보여주는 로그를 생성할 수 있습니다. 이러한 툴은 **OpenStack** 노드에 배포할 때 보호 계층을 제공할 수 있습니다. 프로젝트 사용자는 인스턴스에서 이러한 툴을 실행하려고 할 수도 있습니다.

13장. 프로젝트의 데이터 프라이버시

OpenStack은 다양한 데이터 요구 사항이 있는 프로젝트 간에 멀티 테넌시를 지원하도록 설계되었습니다. 클라우드 운영자는 해당 데이터 개인 정보 취급 방침과 규정을 고려해야 합니다. 이 장에서는 OpenStack 배포를 위한 데이터 리빌리티 및 폐기에 대해 다룹니다.

13.1. 데이터 개인 정보 보호 문제

이 섹션에서는 OpenStack 배포의 데이터 프라이버시에 발생할 수 있는 몇 가지 우려 사항에 대해 설명합니다.

13.1.1. 데이터 상주성

데이터의 기밀성과 격리는 지난 몇 년 동안 클라우드 채택의 주요 장애로 인식되어 왔습니다. 클라우드에서 데이터를 소유하는 사용자와 클라우드 운영자가 궁극적으로 이 데이터의 보호자로서 신뢰할 수 있는지에 대한 우려는 과거의 중요한 문제였습니다.

특정 OpenStack 서비스에는 프로젝트 또는 참조 프로젝트 정보에 속하는 데이터 및 메타데이터에 액세스할 수 있습니다. 예를 들어 OpenStack 클라우드에 저장된 프로젝트 데이터에 다음 항목이 포함될 수 있습니다.

- 오브젝트 스토리지 오브젝트.
- 계산 인스턴스 임시 파일 시스템 스토리지.
- 계산 인스턴스 메모리.
- 블록 스토리지 볼륨 데이터.
- 계산 액세스용 공개 키.
- 이미지 서비스의 가상 시스템 이미지.

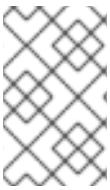
- 인스턴스 스냅샷.
- Compute의 configuration-drive 확장에 전달되는 데이터.

OpenStack 클라우드에서 저장한 메타데이터에는 다음 항목이 포함됩니다(이 목록은 포괄적이지 않음).

- 조직 이름.
- 사용자의 "실무 이름".
- 실행 중인 인스턴스, 버킷, 오브젝트, 볼륨 및 기타 할당량 관련 항목의 수 또는 크기.
- 인스턴스 실행 또는 데이터 저장 시간 수.
- 사용자의 IP 주소.
- 컴퓨팅 이미지 번들을 위해 내부적으로 생성된 개인 키.

13.1.2. 데이터 처리

모범 사례에 따라 운영자는 조직 제어에서 릴리스되기 전에 클라우드 시스템 미디어(디지터 및 비디지컬)를 삭제해야 합니다. 삭제 방법은 특정 보안 도메인과 정보의 민감도에 따라 적절한 수준의 강점과 무결성을 구현해야 합니다.



참고

NIST Special Publication 800-53 버전 4는 이 주제에 대한 특정 관점을 취합니다.

> The sanitization process removes information from the media such that the information cannot be retrieved or reconstructed. Sanitization techniques, including clearing, purging, cryptographic erase, and destruction, prevent the disclosure of information to unauthorized individuals when such media is reused or released for disposal.

클라우드 운영자는 일반 데이터 처리 및 삭제 지침을 개발할 때 다음을 고려해야 합니다(NIST 권장 보안 제어에 따라).

- 미디어 제거 및 폐기 작업 추적, 문서화 및 확인.
- 안전 장비 및 절차를 테스트하여 적절한 성능을 검증합니다.
- 이러한 장치를 클라우드 인프라에 연결하기 전에 이동식 스토리지 장치를 삭제합니다.
- 삭제할 수 없는 클라우드 시스템 미디어 삭제.

결과적으로 OpenStack 배포는 다음 방법을 해결해야 합니다(다른 방법).

- 안전한 데이터 삭제
- 인스턴스 메모리 스크럽
- 블록 스토리지 볼륨 데이터
- 컴퓨팅 인스턴스 임시 스토리지
- 베어 메탈 서버 삭제

13.1.3. 데이터가 안전하게 삭제되지 않음

OpenStack 내에서 일부 데이터는 삭제될 수 있지만 위에 설명된 NIST 표준의 맥락에서 안전하게 지워지지는 않습니다. 이는 일반적으로 위에서 정의한 메타데이터 및 데이터베이스에 저장된 대부분의 또는 모든 메타데이터에 적용할 수 있습니다. 이는 자동 채우기 및 주기적인 자유 공간 삭제를 위한 데이터베이스 및/또는 시스템 구성으로 해결될 수 있습니다.

13.1.4. 인스턴스 메모리 스크럽

인스턴스 메모리의 처리는 다양한 하이퍼바이저에만 해당됩니다. 이 동작은 계산에 정의되지 않지만 일반적으로 하이퍼바이저는 인스턴스를 만들 때 또는 둘 다 삭제할 때 인스턴스를 삭제할 때 메모리를 제거하는 것이 가장 좋습니다.

13.1.5. Cinder 볼륨 데이터 암호화

OpenStack 볼륨 암호화 기능을 사용하는 것이 좋습니다. 이 내용은 Volume Encryption(볼륨 암호화)의 Data Encryption(데이터 암호화) 섹션에서 설명합니다. 이 기능을 사용하면 암호화 키를 안전하게 삭제하여 데이터를 제거합니다. 최종 사용자는 볼륨을 생성하는 동안 이 기능을 선택할 수 있지만 관리자는 먼저 볼륨 암호화 기능을 한 번만 설정해야 합니다.

OpenStack 볼륨 암호화 기능을 사용하지 않으면 일반적으로 다른 접근법을 활성화하기가 더 어렵습니다. 백엔드 플러그인을 사용하는 경우 암호화 또는 비표준 덮어쓰기 솔루션을 수행하는 독립적인 방법이 있을 수 있습니다. OpenStack 블록 스토리지의 플러그인은 다양한 방법으로 데이터를 저장합니다. 많은 플러그인은 벤더 또는 기술에만 해당되는 반면, 다른 플러그인은 파일 시스템(예: LVM 또는 ZFS)을 중심으로 더 DIY 솔루션입니다. 데이터를 안전하게 삭제하는 방법은 플러그인, 벤더 및 파일 시스템마다 다릅니다.

일부 백엔드(예: ZFS)는 데이터 노출을 방지하기 위해 COW(Copy-On-Write)를 지원합니다. 이 경우 쓰기되지 않은 블록에서 읽기는 항상 0을 반환합니다. 다른 백엔드(예: LVM)는 기본적으로 이를 지원하지 않을 수 있으므로 cinder 플러그인은 사용자에게 전달하기 전에 이전에 작성된 블록을 재정의해야 합니다. 선택한 볼륨 백엔드에서 제공하는 보장을 검토하고 제공되지 않은 해당 보장에 대해 사용할 수 있는 수정 사항을 확인하는 것이 중요합니다.

13.1.6. 이미지 서비스 지연 삭제 기능

이미지 서비스에는 삭제 시간이 지연되어 정의된 기간 동안 이미지 삭제가 적용됩니다. 이 동작이 보안 문제가 되는 경우 이 기능을 비활성화하는 것이 좋습니다. glance-api.conf 파일을 편집하고 delay_delete 옵션을 False로 설정하여 이 작업을 수행할 수 있습니다.

13.1.7. 컴퓨팅 소프트 삭제 기능

compute에는 소프트 삭제 기능이 있으므로, 삭제된 인스턴스가 정의된 기간 동안 소프트 삭제 상태가 될 수 있습니다. 이 기간 동안 인스턴스를 복원할 수 있습니다. 소프트 삭제 기능을 비활성화하려면 /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf 파일을 편집하고 reclaim_instance_interval 옵션을 비워 둡니다.

13.1.8. 컴퓨팅 인스턴스의 임시 스토리지

OpenStack Ephemeral disk 암호화 기능은 활성 사용과 데이터가 삭제될 때와 함께 임시 스토리지의 개인 정보 보호 및 격리를 개선하는 수단을 제공합니다. 암호화된 블록 스토리지의 경우와 마찬가지로 암

호화 키를 삭제하면 효과적으로 데이터를 삭제할 수 있습니다.

임시 스토리지를 생성 및 삭제하는 데이터 프라이버시를 제공하는 대체 방법은 선택한 하이퍼바이저와 계산 플러그인에 따라 달라집니다.

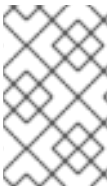
계산용 **libvirt** 드라이버는 파일 시스템 또는 **LVM**에서 임시 스토리지를 직접 유지 관리할 수 있습니다. 파일 시스템 스토리지는 일반적으로 데이터를 제거할 때 덮어쓰지 않습니다. 그러나 더티 확장 영역이 사용자에게 프로비저닝되지 않는다는 보장도 있습니다.

블록 기반 **LVM** 백업 임시 스토리지를 사용하는 경우 정보 공개를 방지하기 위해 계산 소프트웨어에서 블록을 안전하게 지워야 합니다. 이전에는 임시 블록 스토리지 장치와 잘못 삭제된 관련 정보 공개 취약점이 있었습니다.

파일 시스템 스토리지는 더티 확장 영역을 사용자에게 배포할 수 없으므로 **LVM**보다 임시 블록 스토리지 장치에 더 안전한 솔루션입니다. 그러나 사용자 데이터가 삭제되지 않으므로 백업 파일 시스템을 암호화하는 것이 좋습니다.

13.2. 베어 메탈 프로비저닝의 보안 강화

베어 메탈 프로비저닝 인프라의 경우 일반적으로 베이스 보드 관리 컨트롤러(**BMC**) 및 특히 **IPMI**를 강화하는 보안 강화를 고려해야 합니다. 예를 들어 프로비저닝 네트워크 내에서 이러한 시스템을 분리하고, 기본값이 아닌 강력한 암호를 구성하며, 원하지 않는 관리 기능을 비활성화할 수 있습니다. 자세한 내용은 이러한 구성 요소를 강화하는 보안 강화에 대한 벤더의 지침을 참조하십시오.



참고

가능한 경우 기존 레거시 **BMC**보다 **Redfish** 기반 **BMC**를 평가합니다.

13.2.1. 하드웨어 식별

서버를 배포할 때 공격자의 서버와 구분할 수 있는 안정적인 방법이 항상 있는 것은 아닙니다. 이 기능은 어느 정도 하드웨어/**BMC**에 종속될 수 있지만, 일반적으로 서버에 내장된 식별 수단이 없는 것으로 보입니다.

13.3. 데이터 암호화

옵션은 아래에 설명된 **OpenStack** 볼륨 암호화 기능과 같이 디스크에 저장되거나 네트워크를 통해 전송할 때마다 구현자가 프로젝트 데이터를 암호화할 수 있도록 존재합니다. 이는 사용자가 공급업체로 보

내기 전에 자신의 데이터를 암호화하는 일반적인 권장 사항을 넘어선 것입니다.

프로젝트를 대신하여 데이터를 암호화하는 것은 공격자가 프로젝트 데이터에 액세스할 수 있는 공급자가 추정한 위험과 크게 관련이 있습니다. 정부 및 정책별 요구 사항, 프라이빗 계약 또는 공용 클라우드 공급업체의 프라이빗 계약과 관련된 요구 사항이 있을 수 있습니다. 프로젝트 암호화 정책을 선택하기 전에 위험 평가 및 법률적 조언을 받는 것이 좋습니다.

인스턴스별 또는 개체별 암호화는 내림차순, 프로젝트당, 호스트별, 클라우드당 집계를 통해 선호됩니다. 이 권장 사항은 구현의 복잡성과 난이도와 반대입니다. 현재 일부 프로젝트에서는 프로젝트별로 느슨하게 암호화를 구현하는 것이 어렵거나 불가능한 경우도 있습니다. 구현자는 프로젝트 데이터 암호화를 고려해야 합니다.

종종 데이터 암호화는 단순히 키를 폐기하여 프로젝트 및 인스턴스별 데이터를 안정적으로 삭제하는 기능과 긍정적인 관련이 있습니다. 이렇게 하면 신뢰할 수 있고 안전한 방식으로 해당 키를 삭제하는 것이 매우 중요합니다.

사용자의 데이터를 암호화할 수 있는 기회가 있습니다.

- **Object Storage** 오브젝트
- 네트워크 데이터

13.3.1. 볼륨 암호화

OpenStack의 볼륨 암호화 기능은 프로젝트별로 기밀성을 지원합니다. 지원되는 기능은 다음과 같습니다.

- 대시보드 또는 명령줄 인터페이스를 통해 시작되는 암호화된 볼륨 유형의 생성 및 사용
- 암호화를 활성화하고 암호화 알고리즘 및 키 크기와 같은 매개 변수를 선택합니다.
- **iSCSI** 패킷에 포함된 볼륨 데이터가 암호화됨

- 원래 볼륨이 암호화된 경우 암호화된 백업 지원

- 볼륨 암호화 상태의 대시보드 표시. 볼륨이 암호화되고 알고리즘 및 키 크기와 같은 암호화 매개 변수를 포함한다는 표시가 포함됩니다.

- 키 관리 서비스와의 인터페이스

13.3.2. 임시 디스크 암호화

임시 디스크 암호화 기능으로 데이터 프라이버시를 처리합니다. 임시 디스크는 가상 호스트 운영 체제에서 사용하는 임시 작업 공간입니다. 암호화를 사용하지 않으면 이 디스크에서 중요한 사용자 정보에 액세스할 수 있으며 디스크의 마운트를 해제한 후에도 **vestigial** 정보가 유지될 수 있습니다. 다음과 같은 임시 디스크 암호화 기능이 지원됩니다.

13.3.2.1. 암호화된 LVM 임시 디스크 생성 및 사용

참고

계산 서비스는 현재 **LVM** 형식의 임시 디스크 암호화만 지원합니다.

계산 구성 파일(**/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf**)에는 **ephemeral_storage_encryption** 섹션 내에 다음과 같은 기본 매개변수가 있습니다.

- **cipher = aes-xts-plain64 -** 이 필드는 임시 스토리지를 암호화하는 데 사용되는 암호 및 모드를 설정합니다. **AES-XTS**는 특히 디스크 스토리지용으로 **NIST**에서 권장되며, 이름은 **XTS** 암호화 모드를 사용하는 **AES** 암호화에 대해 단축됩니다. 사용 가능한 암호는 커널 지원에 따라 다릅니다. 명령줄에서 **cryptsetup** 벤치마크 를 입력하여 사용 가능한 옵션을 확인하고 벤치마크 결과를 참조하거나 **/proc/crypto**로 이동합니다.

- **enabled = false -** 임시 디스크 암호화를 사용하려면 옵션 설정**: enabled = true**

- **key_size = 512 -** 백엔드 키 관리자의 키 크기 제한이 있을 수 있습니다. **key_size = 256** 을 사용해야 하며, 이 키 크기는 **128**비트의 **AES** 키 크기만 제공해야 합니다. **XTS**는 암호화 키 **AES** 외에도 자체 **"tweak** 키**"**가 필요합니다. 일반적으로 이 키는 하나의 큰 키로 표시됩니다. 이 경우 **512**비트 설정을 사용하면 **XTS**에서 **AES** 및 **256**비트가 **256**비트를 사용합니다. **(NIST** 참조**)**

13.3.2.2. 키 관리 서비스와 상호 작용

키 관리 서비스는 프로젝트별로 임시 디스크 암호화 키를 제공하여 데이터 격리를 지원합니다. 임시 디스크 암호화는 백엔드 키 스토리지에서 지원하여 보안을 강화합니다. 키 관리 서비스를 사용하면 임시 디스크가 더 이상 필요하지 않은 경우 키를 삭제하면 임시 디스크 스토리지 영역을 덮어쓸 수 있습니다.

13.3.3. Object Storage 오브젝트

Object Storage(swift)는 스토리지 노드에서 유휴 상태의 오브젝트 데이터의 선택적 암호화를 지원합니다. 개체 데이터의 암호화는 무단 당사자가 디스크에 대한 물리적 액세스를 얻는 경우 사용자의 데이터를 읽을 위험을 완화하기 위한 것입니다.

유휴 상태의 데이터 암호화는 프록시 서버 **WSGI** 파이프라인에 포함될 수 있는 미들웨어에 의해 구현됩니다. 이 기능은 **Swift** 클러스터 내부이며 **API**를 통해 노출되지 않습니다. 클라이언트는 데이터가 **swift** 서비스로 내부적으로 암호화된다는 것을 인식하지 못합니다. 내부적으로 암호화된 데이터는 **swift API**를 통해 클라이언트로 반환되지 않아야 합니다.

다음 데이터는 **swift**에서 유휴 상태로 유지되는 동안 암호화됩니다.

- 오브젝트 콘텐츠(예: 오브젝트 **PUT** 요청 본문의 콘텐츠)입니다.
- 콘텐츠가 **0**이 아닌 개체의 엔터티 태그(**ETag**)입니다.
- 모든 사용자 지정 사용자 오브젝트 메타데이터 값. 예를 들어 **X-Object-Meta-** 접두사가 지정된 헤더를 **PUT** 또는 **POST** 요청과 함께 사용하여 전송되는 메타데이터입니다.

위 목록에 포함되지 않은 모든 데이터 또는 메타데이터는 다음을 포함하여 암호화되지 않습니다.

- 계정, 컨테이너 및 오브젝트 이름
- 계정 및 컨테이너 사용자 정의 사용자 메타데이터 값
- 모든 사용자 정의 사용자 메타데이터 이름

- 오브젝트 **Content-Type** 값
- 오브젝트 크기
- 시스템 메타데이터

13.3.4. 블록 스토리지 성능 및 백엔드

운영 체제를 활성화하면 현재 **Intel** 및 **AMD** 프로세서에서 사용 가능한 하드웨어 가속 기능을 사용하여 **OpenStack** 볼륨 암호화 성능을 향상시킬 수 있습니다. **OpenStack** 볼륨 암호화 기능과 **OpenStack Ephemeral Disk Encryption** 기능은 모두 **dm-crypt** 를 사용하여 볼륨 데이터를 보호합니다. **dm-crypt** 는 **Linux** 커널 버전 **2.6** 이상에서 투명한 디스크 암호화 기능입니다. 하드웨어 가속 기능을 사용하면 두 암호화 기능의 성능에 미치는 영향이 최소화됩니다.

13.3.5. 네트워크 데이터

컴퓨팅 노드의 프로젝트 데이터는 **IPsec** 또는 기타 터널을 통해 암호화할 수 있습니다. 이 방법은 **OpenStack**에서 일반 또는 표준이 아닌 동기부여 및 관심 있는 구현자가 사용할 수 있는 옵션입니다. 마찬가지로, 암호화된 데이터는 네트워크를 통해 전송되므로 계속 암호화됩니다.

13.4. 키 관리

프로젝트 데이터 프라이버시에 대해 자주 언급된 문제를 해결하기 위해 **OpenStack** 커뮤니티에 데이터 암호화가 보다 유비쿼터스되도록 하는 중요한 관심이 있습니다. 최종 사용자가 클라우드에 저장하기 전에 데이터를 암호화하는 것은 비교적 쉽습니다. 이 경로는 미디어 파일, 데이터베이스 아카이브와 같은 프로젝트 객체를 위한 실행 가능한 경로입니다. 클라이언트 측 암호화는 나중에 사용하기 위해 데이터를 해독하기 위해 키를 표시하는 등 클라이언트 상호 작용이 필요한 가상화 기술에서 보유하는 데이터를 암호화하는 데 사용됩니다.

Barbican은 사용자가 키 관리를 부담하지 않고도 프로젝트에 데이터를 보다 원활하게 암호화하고 액세스할 수 있도록 도움을 줄 수 있습니다. **OpenStack**의 일부로 암호화 및 키 관리 서비스를 제공하여 데이터 처리에 필요한 보안 채택을 용이하게 하며, 개인 정보 보호나 데이터 오용에 대한 고객의 우려를 해소할 수 있습니다.

볼륨 암호화 및 임시 디스크 암호화 기능은 키 관리 서비스(예: **barbican**)를 사용하여 키 생성 및 보안 강화를 위해 사용됩니다.

14장. 인스턴스 보안 관리

가상화 환경에서 인스턴스를 실행할 때의 이점 중 하나는 일반적으로 베어 메탈에 배포할 때 사용할 수 없는 보안 제어의 새로운 기회입니다. **OpenStack** 배포를 위해 개선된 정보 보장을 제공하는 가상화 스택에 특정 기술을 적용할 수 있습니다. 강력한 보안 요구 사항이 있는 운영자는 이러한 기술을 배포하는 것을 고려하려고 할 수 있지만 모든 상황에는 모두 적용할 수 있는 것은 아닙니다. 정규화된 비즈니스 요구 사항으로 인해 기술이 클라우드에서 사용하기 위해 제외될 수 있는 경우도 있습니다. 마찬가지로 일부 기술은 시스템 사용자에게 바람직하지 않을 수 있는 실행 상태와 같은 인스턴스 데이터를 검사합니다.

이 장에서는 이러한 기술과 인스턴스 또는 기본 노드의 보안을 강화하는 데 사용할 수 있는 상황을 설명합니다. 또한 데이터 패스스루, 인트로스펙션 또는 엔트로피 소스를 포함할 수 있는 개인 정보 보호 문제도 강조되어 있습니다.

14.1. 인스턴스에 엔트로피 제공

이 장에서는 *엔트로피* 라는 용어를 사용하여 인스턴스에서 사용할 수 있는 임의 데이터의 품질과 소스를 나타냅니다. 일반적으로 암호화 기술은 높은 품질의 엔트로피 풀을 만들어야 하는 임의성에 의존합니다. 이러한 작업을 지원하기에 일반적으로 인스턴스에서 충분한 엔트로피를 얻기가 어렵습니다. 이를 엔트로피 부족이라고 합니다. 이 조건은 인스턴스에 관련이 없는 것으로 표시될 수 있습니다. 예를 들어 느린 부팅 시간은 인스턴스가 **SSH** 키 생성을 기다리고 있기 때문일 수 있습니다. 또한 이 조건으로 인해 사용자가 인스턴스 내에서 품질이 낮은 엔트로피 소스를 사용할 위험이 있으므로 클라우드에서 실행되는 애플리케이션의 보안은 전반적으로 줄어듭니다.

다행히도 높은 품질의 엔트로피 소스를 인스턴스에 제공하여 이러한 문제를 해결할 수 있습니다. 이 작업은 인스턴스를 지원하기에 클라우드에 충분한 하드웨어 임의 번호 생성기(**HRNG**)를 보유하여 수행할 수 있습니다. 이 경우 도메인에 따라 충분히 제한됩니다. 일상 작업에서 최신 **HRNG**은 **50-100** 계산 노드를 지원하기에 충분한 엔트로피를 생성할 수 있습니다. **Intel** 아이비어 브리지 및 최신 프로세서에서 사용할 수 있는 **RdRand** 명령과 같은 높은 대역폭 **HRNG**은 더 많은 노드를 처리할 수 있었습니다. 지정된 클라우드의 경우 설계자는 애플리케이션 요구 사항을 이해하여 엔트로피를 충분히 사용할 수 있는지 확인해야 합니다.

Virtio RNG은 기본적으로 **/dev/random** 을 엔트로피 소스로 사용하는 임의 숫자 생성기입니다. 하드웨어 **RNG** 또는 엔트로피 수집 데몬(**EGD**)과 같은 도구를 사용하여 배포를 통해 엔트로피를 적절하게 배포하는 방법을 제공하도록 구성할 수도 있습니다. **hw_rng metadata** 속성을 사용하여 인스턴스 생성 시 **Virtio RNG**를 활성화할 수 있습니다.

14.2. 노드에 인스턴스 예약

인스턴스를 만들기 전에 이미지 인스턴스화를 위한 호스트를 선택해야 합니다. 이 선택은 계산 및 볼륨 요청을 디스패치하는 방법을 결정하는 **nova-scheduler** 에서 수행합니다.

FilterScheduler 는 다른 스케줄러가 있지만 계산용 기본 스케줄러입니다. 이 기능은 *필터 힌트* 와 협력하여 인스턴스를 시작해야 하는 위치를 결정합니다. 이 호스트 선택 프로세스를 통해 관리자는 다양한 보안 및 규정 준수 요구 사항을 충족할 수 있습니다. 데이터 격리가 주요 문제인 경우 가능한 경우 가능한 경우 프로젝트 인스턴스를 동일한 호스트에 제공하도록 선택할 수 있습니다. 반대로, 가용성 또는 내결함성을 위해 가능한 한 많은 다른 호스트에 인스턴스를 배치할 수 있습니다.

필터 스케줄러는 다음 기본 카테고리에 속합니다.

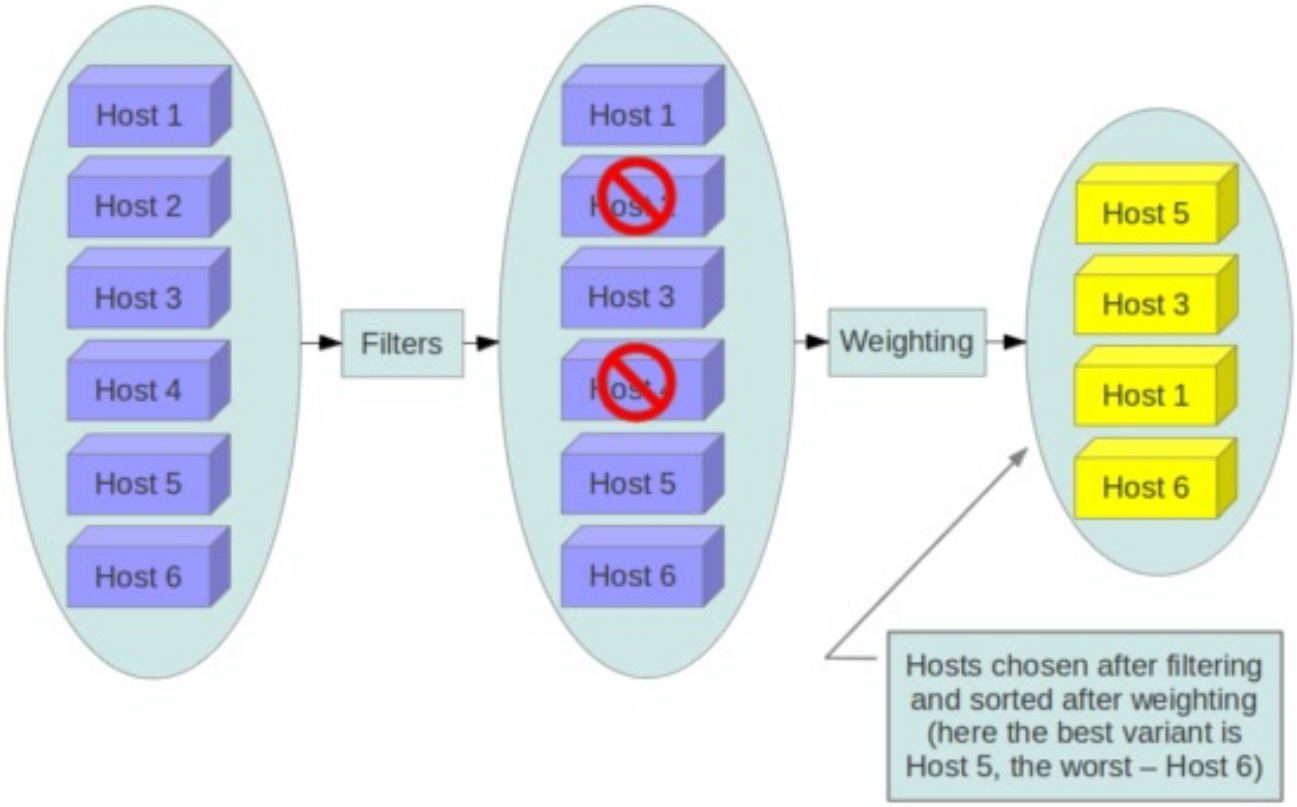
- ***Resource based filter(리소스 기반 필터)*** **-** 하이퍼바이저 호스트 세트의 시스템 리소스 사용량에 따라 인스턴스 배치를 결정하고 **RAM, IO** 또는 **CPU** 사용률과 같은 무료 또는 사용되는 속성에서 트리거할 수 있습니다.

- *이미지 기반 필터* **-** VM의 운영 체제 또는 사용된 이미지 유형과 같이 사용되는 이미지 메타데이터를 기반으로 인스턴스 생성을 위임합니다.

- *환경 기반 필터* **-** 특정 **IP** 범위 내에서, 가용 영역 전체 또는 다른 인스턴스와 동일한 호스트에 같은 외부 세부 정보를 기반으로 인스턴스 배치를 결정합니다.

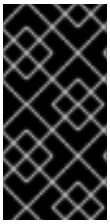
- *사용자 지정 기준* **-** 신뢰 또는 메타데이터 구문 분석과 같은 사용자 또는 관리자가 제공한 기준에 따라 인스턴스 생성을 위임합니다.

여러 필터를 한 번에 적용할 수 있습니다. 예를 들어 **ServerGroupAffinity** 필터는 인스턴스가 특정 호스트 집합의 멤버에 생성되고 **ServerGroupAntiAffinity** 필터는 인스턴스가 다른 특정 호스트 집합에서 생성되지 않았는지 확인합니다. 이러한 두 필터는 일반적으로 동시에 활성화되며 지정된 속성의 값에 대한 각 검사와 충돌할 수 없으며 동시에 **true**일 수 없습니다.



참고

DiskFilter 필터는 디스크 공간을 초과 구독할 수 있습니다. 일반적으로 문제가 되지 않지만 씬 프로비저닝된 스토리지 장치에 문제가 될 수 있습니다. 이 필터는 잘 테스트된 할당량과 함께 사용해야 합니다. 이 기능은 사용되지 않으며 **Red Hat OpenStack Platform 12** 이후에는 사용할 수 없습니다.



중요

사용자가 제공하거나 조작할 수 있는 오브젝트(예: 메타데이터)를 구문 분석하는 필터를 비활성화하는 것이 좋습니다.

14.3. 신뢰할 수 있는 이미지 사용

클라우드 환경에서 사용자는 업로드된 사전 설치된 이미지 또는 이미지를 사용합니다. 두 경우 모두 사용자는 사용 중인 이미지가 로 변조되지 않았는지 확인할 수 있어야 합니다. 이미지를 확인하는 기능은 보안을 위한 기본적인 필수 요소입니다. 이미지 소스에서 사용되는 대상까지 신뢰할 수 있는 체인이 필요합니다. 이 작업은 신뢰할 수 있는 소스에서 얻은 이미지에 서명하고 사용하기 전에 서명을 확인하여 수행할

수 있습니다. 아래에서 확인된 이미지를 가져오고 생성하는 다양한 방법에 대해 설명한 다음 이미지 서명 확인 기능에 대한 설명이 나와 있습니다.

14.3.1. 이미지 생성

OpenStack 설명서에서는 이미지를 만들고 이미지 서비스에 업로드하는 방법에 대한 지침을 제공합니다. 또한 게스트 운영 체제를 설치하고 강화하는 프로세스가 있다고 가정합니다. 다음 항목은 이미지를 **OpenStack**으로 전송하는 방법에 대한 추가 지침을 제공합니다. 이미지를 가져오는 데는 다양한 옵션이 있습니다. 각각에는 이미지의 출처를 확인하는 데 도움이 되는 특정 단계가 있습니다.

- 옵션 **1:** 신뢰할 수 있는 소스에서 부팅 미디어를 가져옵니다. 예를 들어 공식 **Red Hat** 소스에서 이미지를 다운로드한 다음 추가 체크섬 유효성 검사를 수행할 수 있습니다.
- 옵션 **2: OpenStack** 가상 시스템 이미지 가이드 사용. 이 경우 조직 **OS** 강화 지침을 준수하려고 합니다.
- 옵션 **3:** 자동화된 이미지 빌더 사용. 다음 예제에서는 **Oz** 이미지 빌더를 사용합니다. **OpenStack** 커뮤니티는 최근에 **disk-image-builder** 라는 새로운 도구를 생성했으며 아직 보안 평가를 받지 않았습니다.

이 예제에서 **RHEL 6 CCE-26976-1** 은 **Oz** 내에서 **NIST 800-53** 섹션 **AC-19(d)**를 구현하는 데 도움이 됩니다.

```
<template>
<name>centos64</name>
<os>
  <name>RHEL-6</name>
  <version>4</version>
  <arch>x86_64</arch>
  <install type='iso'>
  <iso>http://trusted_local_iso_mirror/isos/x86_64/RHEL-6.4-x86_64-bin-DVD1.iso</iso>
  </install>
  <rootpw>CHANGE THIS TO YOUR ROOT PASSWORD</rootpw>
</os>
<description>RHEL 6.4 x86_64</description>
<repositories>
  <repository name='epel-6'>
  <url>http://download.fedoraproject.org/pub/epel/6/$basearch</url>
  <signed>no</signed>
  </repository>
</repositories>
<packages>
  <package name='epel-release'/>
  <package name='cloud-utils'/>
  <package name='cloud-init'/>
```

```
</packages>
<commands>
  <command name='update'>
  yum update
  yum clean all
  sed -i '/^HWADDR/d' /etc/sysconfig/network-scripts/ifcfg-eth0
  echo -n > /etc/udev/rules.d/70-persistent-net.rules
  echo -n > /lib/udev/rules.d/75-persistent-net-generator.rules
  chkconfig --level 0123456 autofs off
  service autofs stop
  </command>
</commands>
</template>
```

수동 이미지 빌드 프로세스는 복잡하고 오류가 발생하기 쉬운 것을 피하는 것이 좋습니다. 또한 이미지 빌드를 위해 **Oz**와 같은 자동 시스템 또는 부팅 후 이미지 강화를 위한 구성 관리 유틸리티(예: **Chef** 또는 **Puppet**)를 사용하면 일관된 이미지를 생성할 수 있으며, 시간 경과에 따라 기본 이미지의 규정 준수를 추적할 수 있습니다.

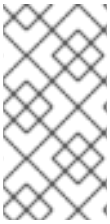
공용 클라우드 서비스를 서브스크립션하는 경우 클라우드 공급자에게 기본 이미지를 생성하는 데 사용되는 프로세스의 개요를 확인해야 합니다. 공급자가 자체 이미지를 업로드할 수 있는 경우 인스턴스를 생성하는 데 사용하기 전에 이미지가 수정되지 않았는지 확인해야 합니다. 이렇게 하려면 _ 이미지 서명 확인에 있는 다음 섹션을 참조하거나 서명을 사용할 수 없는 경우 다음 단락을 참조하십시오.

이미지 서비스(**glance**)는 이미지를 노드의 계산 서비스에 업로드하는 데 사용합니다. 이 전송은 **TLS**를 통해 더욱 강화되어야 합니다. 이미지가 노드에 있으면 기본 체크섬으로 확인되며, 해당 디스크는 시작 중인 인스턴스의 크기에 따라 확장됩니다. 나중에 이 노드에서 동일한 인스턴스 크기로 동일한 이미지가 시작되면 동일한 확장된 이미지에서 시작됩니다. 이 확장된 이미지는 시작하기 전에 기본적으로 다시 확인되지 않으므로 변조를 겪은 위험이 있습니다. 결과 이미지에서 파일을 수동으로 검사하지 않는 한 사용자는 변조를 알지 못합니다. 이 문제를 완화하려면 이미지 서명 확인 항목에 대한 다음 섹션을 참조하십시오.

14.3.2. 이미지 서명 확인

이미지 서명과 관련된 특정 기능을 **OpenStack**에서 사용할 수 있습니다. **Red Hat OpenStack Platform 13**부터 이미지 서비스는 이러한 서명된 이미지를 확인할 수 있으며 전체 신뢰 체인을 제공하기 위해 계산 서비스에는 이미지 부팅 전에 이미지 서명 확인을 수행할 수 있는 옵션이 있습니다. 이미지를 부팅하기 전에 서명된 이미지가 변경되지 않았는지 확인합니다. 이 기능을 활성화하면 이미지의 무단 수정(예: 맬웨어 또는 루트킷을 포함하도록 이미지 수정)을 탐지할 수 있습니다.

/var/lib/config-data/puppet-generated/nova _libvirt/etc/nova/nova.conf 파일에서 **verify_glance_signatures** 플래그를 **True** 로 설정하여 인스턴스 서명 확인을 활성화할 수 있습니다. 활성화되면 **Compute** 서비스는 **Glance**에서 검색할 때 서명된 인스턴스의 유효성을 자동으로 검사합니다. 이 확인에 실패하면 부팅 프로세스가 시작되지 않습니다.



참고

이 기능이 활성화되면 서명되지 않은 이미지(서브스크립트되지 않은 이미지)가 없는 이미지도 확인에 실패하고 부팅 프로세스가 시작되지 않습니다.

14.4. 인스턴스 마이그레이션

OpenStack 및 기본 가상화 계층에서는 **OpenStack** 노드 간 이미지를 실시간 마이그레이션하여 인스턴스 가동 중단 없이 컴퓨팅 노드의 롤링 업그레이드를 원활하게 수행할 수 있습니다. 그러나 실시간 마이그레이션에도 상당한 위험 요소가 있습니다. 관련 위험을 이해하기 위해 실시간 마이그레이션 중에 수행되는 상위 수준 단계는 다음과 같습니다.

1. 대상 호스트에서 인스턴스 시작
2. 전송 메모리
3. 게스트를 중지하고 디스크 동기화
4. 상태 전송
5. 게스트 시작

참고

콜드 마이그레이션, 크기 조정 및 보류와 같은 특정 작업은 모두 일정량의 인스턴스 데이터를 네트워크 전반에서 다른 서비스로 전송할 수 있습니다.

14.4.1. 실시간 마이그레이션 위험

실시간 마이그레이션 프로세스의 다양한 단계에서 인스턴스 런타임 메모리 및 디스크의 콘텐츠가 네트워크를 통해 일반 텍스트로 전송됩니다. 따라서 실시간 마이그레이션을 사용할 때 해결해야 하는 여러 위험이 있습니다. 다음 완전하지 않은 목록은 이러한 위험 중 일부를 자세히 설명합니다.

- 서비스 거부 **(DoS):** 마이그레이션 프로세스 중에 오류가 발생하면 인스턴스가 손실될 수 있습니다.

- 데이터 노출: 메모리 또는 디스크 전송을 안전하게 처리해야 합니다.

- 데이터 조작: 메모리 또는 디스크 전송이 안전하게 처리되지 않으면 공격자가 마이그레이션 중에 사용자 데이터를 조작할 수 있습니다.

- 코드 주입: 메모리 또는 디스크 전송이 안전하게 처리되지 않으면 공격자는 디스크 또는 메모리에서 실행 파일을 조작할 수 있습니다.

14.4.2. 실시간 마이그레이션 완화

실시간 마이그레이션과 관련된 위험을 완화하는 데 도움이 되는 여러 가지 방법이 있습니다. 이러한 내용은 다음 섹션에서 설명합니다.

14.4.2.1. 실시간 마이그레이션 비활성화

현재 **OpenStack**에서 실시간 마이그레이션은 기본적으로 활성화되어 있습니다. 실시간 마이그레이션은 기본적으로 관리자 전용 작업이므로 사용자는 이 작업을 시작할 수 없으며 관리자만(신뢰할 수 있음) 관리자만 시작할 수 없습니다. **nova policy.json** 파일에 다음 행을 추가하여 실시간 마이그레이션을 비활성화할 수 있습니다.

```
"compute_extension:admin_actions:migrate": "!",
"compute_extension:admin_actions:migrateLive": "!",
```

또는 **TCP** 포트 **49152~49261** 을 차단할 때 실시간 마이그레이션이 실패 하거나 **nova** 사용자가 계산 호스트 간에 암호 없는 **SSH** 액세스 권한이 없는지 확인할 수 있습니다.

실시간 마이그레이션을 위한 **SSH** 구성이 상당히 잠겨 있습니다. 새 사용자가 생성되고 **(nova_migration) SSH** 키는 해당 사용자로 제한되며 허용 목록에 지정된 네트워크에서만 사용할 수 있습니다. 그런 다음 래퍼 스크립트는 실행할 수 있는 명령을 제한합니다(예: **libvirt** 소켓의 **netcat**).

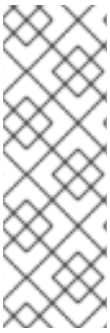
14.4.2.2. 마이그레이션 네트워크

실시간 마이그레이션 트래픽은 실행 중인 인스턴스의 디스크 및 메모리를 일반 텍스트로 전송하며 현재 기본적으로 내부 **API** 네트워크에 호스팅됩니다.

14.4.2.3. 암호화된 실시간 마이그레이션

실시간 마이그레이션을 계속 활성화하기 위한 충분한 요구 사항(예: 업그레이드)이 있는 경우 **libvirtd**는 실시간 마이그레이션에 대해 암호화된 터널을 제공할 수 있습니다. 그러나 이 기능은 **OpenStack** 대시보드 또는 **nova-client** 명령에 노출되지 않으며 **libvirtd**의 수동 구성을 통해서만 액세스할 수 있습니다. 그러면 실시간 마이그레이션 프로세스가 다음과 같은 상위 수준 단계로 변경됩니다.

1. 인스턴스 데이터는 하이퍼바이저에서 **libvirtd**로 복사됩니다.
2. 암호화된 터널은 소스 및 대상 호스트의 **libvirtd** 프로세스 간에 생성됩니다.
3. 대상 **libvirtd** 호스트는 인스턴스를 기본 하이퍼바이저에 다시 복사합니다.



참고

Red Hat OpenStack Platform 13의 경우 권장되는 방법은 터널링된 마이그레이션을 사용하는 것입니다. 이 마이그레이션은 기본적으로 **Ceph**를 백엔드로 사용할 때 활성화됩니다. 자세한 내용은 https://docs.openstack.org/nova/queens/configuration/config.html#libvirt.live_migration_tunnelled 의 내용을 참조하십시오.

14.5. 모니터링, 경고 및 보고

인스턴스는 여러 호스트에 복제할 수 있는 서버 이미지입니다. 따라서 물리적 호스트와 가상 호스트 간에 로깅을 유사한 방식으로 적용하는 것이 좋습니다. 호스트 및 데이터에 대한 액세스 이벤트, 사용자 추가 및 제거, 권한 변경 사항 등의 운영 체제 및 애플리케이션 이벤트를 로깅해야 합니다. 로그 이벤트를 수집하고 분석을 위해 서로 연결한 다음 참조 또는 추가 작업을 위해 저장하는 로그 집계기로 결과를 내보내는 것이 좋습니다. 이 작업을 수행하는 일반적인 도구 중 하나는 **ELK** 스택 또는 **Elasticsearch, Logstash** 및 **Kibana**입니다.

참고

이러한 로그는 정기적으로 검토하거나 네트워크 운영 센터(**NOC**)에서 수행하는 실시간 보기 내에서 모니터링해야 합니다.

작업을 위해 응답자에게 보내지는 경고를 트리거할 이벤트를 추가로 결정해야 합니다.

자세한 내용은 **Monitoring Tools Configuration Guide**를 참조하십시오.

14.5.1. 업데이트 및 패치

하이퍼바이저는 독립 가상 시스템을 실행합니다. 이 하이퍼바이저는 운영 체제에서 실행되거나 하드웨어(베어 메탈이라고 함)에서 직접 실행할 수 있습니다. 하이퍼바이저 업데이트는 가상 머신으로 전파되지 않습니다. 예를 들어 배포에서 KVM을 사용하고 있고 CentOS 가상 시스템 세트가 있는 경우 KVM에 대한 업데이트는 CentOS 가상 시스템에서 실행 중인 모든 것을 업데이트하지 않습니다.

가상 시스템의 강화, 배포 및 지속적인 기능을 담당하는 가상 시스템의 소유권을 소유자에게 할당하는 것이 좋습니다. 또한 업데이트를 정기적으로 배포하는 계획과 프로덕션과 유사한 환경에서 업데이트를 테스트하는 계획도 있어야 합니다.

14.5.2. 방화벽 및 인스턴스 프로필

가장 일반적인 운영 체제에는 추가 보안 계층을 위한 호스트 기반 방화벽이 포함됩니다. 인스턴스는 가능한 한 적은 수의 애플리케이션(최대 단일 용도 인스턴스)을 실행해야 하는 반면, 인스턴스에서 실행 중인 모든 애플리케이션을 프로파일링하여 애플리케이션이 액세스해야 하는 시스템 리소스, 실행에 필요한 최저 권한 수준 및 예상 네트워크 트래픽이 가상 시스템에 들어오고 들어오는지 확인해야 합니다. 예상 트래픽은 SSH 또는 RDP와 같은 필요한 로깅 및 관리 통신과 함께 허용된 트래픽(또는 허용 목록에 포함)으로 호스트 기반 방화벽에 추가해야 합니다. 다른 모든 트래픽은 방화벽 구성에서 명시적으로 거부되어야 합니다.

Linux 인스턴스에서 위의 애플리케이션 프로필을 audit2allow 와 같은 도구와 함께 사용하여 대부분의 Linux 배포판에서 중요한 시스템 정보를 추가로 보호하는 SELinux 정책을 구축할 수 있습니다. SELinux는 사용자, 정책 및 보안 컨텍스트의 조합을 사용하여 애플리케이션을 실행하는 데 필요한 리소스를 구분하고 필요하지 않은 다른 시스템 리소스와 분할합니다.

참고

Red Hat OpenStack Platform에는 기본적으로 SELinux가 활성화되어 있으며 OpenStack 서비스에 맞게 사용자 지정된 정책을 사용할 수 있습니다. 필요에 따라 이러한 정책을 정기적으로 검토하는 것이 좋습니다.

14.5.2.1. 보안 그룹

OpenStack에서는 호스트와 네트워크 모두에 대한 보안 그룹을 제공하여 지정된 프로젝트의 인스턴스에 대한 심층적인 보안을 추가합니다. 이러한 방화벽은 포트, 프로토콜 및 주소를 기반으로 들어오는 트래픽을 허용하거나 거부하는 호스트 기반 방화벽과 유사합니다. 그러나 보안 그룹 규칙은 들어오는 트래픽에만 적용되지만 호스트 기반 방화벽 규칙을 들어오고 나가는 트래픽에 모두 적용할 수 있습니다. 호스트 및 네트워크 보안 그룹 규칙이 합법적인 트래픽을 충돌하고 거부할 수도 있습니다. 사용 중인 네트워킹에 맞게 보안 그룹이 올바르게 구성되었는지 확인하는 것이 좋습니다. 자세한 내용은 이 가이드의 보안 그룹을 참조하십시오.

참고

특히 비활성화할 필요가 없는 경우 보안 그룹과 포트 보안이 활성화된 상태로 유지해야 합니다. 방어에 대한 심도 있는 접근 방식을 구축하려면 세분화된 규칙을 인스턴스에 적용하는 것이 좋습니다.

14.5.3. 인스턴스 콘솔에 액세스

기본적으로 인스턴스의 콘솔에는 가상 콘솔을 통해 원격으로 액세스할 수 있습니다. 이 기능은 문제 해결에 유용할 수 있습니다. **Red Hat OpenStack Platform**은 원격 콘솔 액세스를 위해 **VNC**를 사용합니다.

- 방화벽 규칙을 사용하여 **VNC** 포트를 잠그는 것이 좋습니다. 기본적으로 **nova_vnc_proxy**는 **6080** 및 **13080** 을 사용합니다.

- **VNC** 트래픽이 **TLS**를 통해 암호화되는지 확인합니다. **director** 기반 배포의 경우 **UseTLSTransportForVnc** 로 시작합니다.

14.5.4. 인증서 주입

인스턴스에 **SSH**를 실행해야 하는 경우 생성 시 필요한 **SSH** 키를 인스턴스에 자동으로 삽입하도록 **Compute**를 구성할 수 있습니다.

자세한 내용은 https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html-single/instances_and_images_guide/#section-create-images를 참조하십시오.

15장. 메시지 대기열

메시지 대기열 서비스는 OpenStack의 프로세스 간 통신을 용이하게 합니다. 이러한 메시지는 서비스 백엔드를 사용하여 수행됩니다.

- RabbitMQ - Red Hat OpenStack Platform은 기본적으로 RabbitMQ를 사용합니다.
- Qpid

RabbitMQ 및 Qpid는 모두 AMQP(Advanced Message Queuing Protocol) 프레임워크로, 피어 투 피어 통신에 메시지 대기열을 제공합니다. 대기열 구현은 일반적으로 대기열 서버의 중앙 집중식 또는 분산 풀로 배포됩니다.

메시지 대기열은 OpenStack 배포에서 명령 및 제어 기능을 효과적으로 지원합니다. 큐에 대한 액세스가 허용되면 추가 권한 부여 확인이 수행되지 않습니다. 큐를 통해 액세스할 수 있는 서비스는 실제 메시지 페이로드 내의 컨텍스트 및 토큰의 유효성을 검사합니다. 그러나 토큰이 잠재적으로 재생 가능하며 인프라의 다른 서비스를 인증할 수 있으므로 토큰 만료 날짜를 기록해야 합니다.

OpenStack은 메시지 서명과 같은 메시지 수준의 신뢰를 지원하지 않습니다. 따라서 메시지 전송 자체를 보안하고 인증해야 합니다. HA(고가용성) 구성의 경우 대기열 간 인증 및 암호화를 수행해야 합니다.

15.1. 메시징 보안

이 섹션에서는 OpenStack에 사용되는 가장 일반적인 세 가지 메시지 대기열 솔루션의 보안 강화 접근 방식에 대해 설명합니다. RabbitMQ 및 Qpid.

15.2. 메시징 전송 보안

AMQP 기반 솔루션(Qpid 및 RabbitMQ)은 TLS를 사용하여 전송 수준 보안을 지원합니다.

메시지 큐에 대해 전송 수준 암호화를 활성화하는 것이 좋습니다. 메시징 클라이언트 연결에 TLS를 사용하면 전송 시 변조 및 도청하는 통신이 메시징 서버로 보호됩니다. 일반적으로 TLS를 구성하는 방법에 대한 지침은 아래에 설명되어 있습니다. Qpid 및 RabbitMQ. 메시징 서버가 클라이언트 연결을 확인하는 데 사용하는 신뢰할 수 있는 CA(인증 기관) 번들을 구성할 때 노드에 사용되는 CA(내부 관리 CA)로만 제한하는 것이 좋습니다. 신뢰할 수 있는 CA 번들에서 인증할 클라이언트 인증서를 확인하고 TLS 연결 설정의 클라이언트-서버 확인 단계를 전달합니다.

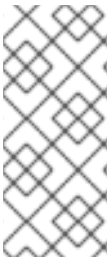
참고

인증서 및 키 파일을 설치하는 경우(예: **chmod 0600** 사용) 파일 권한이 제한되었는지 확인하고 메시징 서버의 다른 프로세스 및 사용자의 무단 액세스를 방지하기 위해 메시징 서버 데몬 사용자로 소유권이 제한됩니다.

15.2.1. RabbitMQ 서버 SSL 구성

다음 행은 시스템 전체 **RabbitMQ** 구성 파일(일반적으로 **/etc/rabbitmq/rabbitmq.config**)에 추가되어야 합니다.

```
[
  {rabbit, [
    {tcp_listeners, [] },
    {ssl_listeners, [{"<IP address or hostname of management network interface>", 5671}] },
    {ssl_options, [{cacertfile,"/etc/ssl/cacert.pem"},
                   {certfile,"/etc/ssl/rabbit-server-cert.pem"},
                   {keyfile,"/etc/ssl/rabbit-server-key.pem"},
                   {verify,verify_peer},
                   {fail_if_no_peer_cert,true}]}
    ]}
].
```



참고

SSL이 아닌 포트에서 수신하지 못하도록 **tcp_listeners** 옵션이 **[]** 로 설정되어 있습니다. **ssl_listeners** 옵션은 서비스에 대한 관리 네트워크에서만 수신 대기하도록 제한해야 합니다.

15.3. 대기열 인증 및 액세스 제어

RabbitMQ 및 **Qpid**는 큐에 대한 액세스를 제어하기 위한 인증 및 액세스 제어 메커니즘을 제공합니다.

SASL(Simple Authentication and Security Layer)은 인터넷 프로토콜에서 인증 및 데이터 보안을 위한 프레임워크입니다. **RabbitMQ**와 **Qpid**는 모두 **SASL** 및 기타 플러그형 인증 메커니즘을 제공하므로 간단한 사용자 이름 및 암호는 인증 보안을 강화할 수 있습니다. **RabbitMQ**는 **SASL**을 지원하지만 **OpenStack**에서 지원은 현재 특정 **SASL** 인증 메커니즘을 요청할 수 없습니다. **OpenStack**에서 **RabbitMQ** 지원을 사용하면 암호화되지 않은 연결 또는 사용자 이름 및 암호에 대한 사용자 이름 및 암호 인증을 **X.509** 클라이언트 인증서와 함께 사용하여 보안 **TLS** 연결을 설정할 수 있습니다.

메시징 대기열에 대한 클라이언트 연결과 가능한 경우(현재 **Qpid**)가 **X.509** 클라이언트 인증서로 인증을 수행하도록 모든 **OpenStack** 서비스 노드에서 **X.509** 클라이언트 인증서를 구성하는 것이 좋습니다.

사용자 이름과 암호를 사용하는 경우 큐에 대한 액세스의 세밀한 감사를 위해 서비스별 및 노드를 생성해야 합니다.

배포하기 전에 대기열 서버에서 사용하는 TLS 라이브러리를 살펴보십시오. Qpid는 Mozilla의 NSS 라이브러리를 사용하는 반면 RabbitMQ는 OpenSSL을 사용하는 Erlang의 TLS 모듈을 사용합니다.

15.3.1. RabbitMQ에 대한 OpenStack 서비스 구성

이 섹션에서는 OpenStack 서비스의 일반적인 RabbitMQ 구성에 대해 설명합니다.

```
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_kombu
rabbit_use_ssl = True
rabbit_host = RABBIT_HOST
rabbit_port = 5671
rabbit_user = compute01
rabbit_password = RABBIT_PASS
kombu_ssl_keyfile = /etc/ssl/node-key.pem
kombu_ssl_certfile = /etc/ssl/node-cert.pem
kombu_ssl_ca_certs = /etc/ssl/cacert.pem
```

참고

RABBIT_PASS 를 적절한 암호로 교체합니다.

15.3.2. Qpid용 OpenStack 서비스 구성

이 섹션에서는 OpenStack 서비스의 일반적인 Qpid 구성에 대해 설명합니다.

```
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_qpid
qpid_protocol = ssl
qpid_hostname = <IP or hostname of management network interface of messaging server>
qpid_port = 5671
qpid_username = compute01
qpid_password = QPID_PASS
```

참고

QPID_PASS 를 적절한 암호로 바꿉니다.

선택적으로 **Qpid**와 함께 **SASL**을 사용하는 경우 다음을 추가하여 사용할 **SASL** 메커니즘을 지정합니다.

```
qpid_sasl_mechanisms = <space separated list of SASL mechanisms to use for auth>
```

15.4. 메시지 큐 프로세스 격리 및 정책

각 프로젝트는 메시지를 보내고 사용하는 여러 서비스를 제공합니다. 메시지를 보내는 각 바이너리는 대기열에서 응답하는 경우에만 메시지를 사용합니다.

메시지 대기열 서비스 프로세스는 시스템의 다른 프로세스와 서로 분리되어야 합니다.

15.4.1. 네임스페이스

Neutron 및 **OVS(Open vSwitch)** 네트워킹은 네임스페이스 내에서 실행되지만, **vswitchd, libvirtd, QEMU**를 포함하여 특정 **OVS** 호스트 서비스는 실행되지 않습니다. 컨테이너화된 컨트롤 플레인을 실행하는 경우 모든 컨트롤 플레인 서비스는 기본적으로 네트워크 네임스페이스 내에서 실행됩니다. 네트워크 네임스페이스는 컴퓨팅 하이퍼바이저에서 실행되는 모든 서비스에 권장됩니다(**AMQ** 서비스를 실행하기 때문에). 이 방법은 인스턴스와 관리 네트워크 간의 네트워크 트래픽 브리징을 방지하는 데 도움이 될 수 있습니다.

16장. API 끝점 강화

OpenStack 클라우드와 관련된 프로세스는 API 엔드포인트를 쿼리하는 것으로 시작됩니다. 퍼블릭 및 프라이빗 엔드포인트에는 여러 가지 문제가 있지만, 이러한 자산은 손상된 경우 상당한 위험을 초래할 수 있는 높은 가치 자산입니다.

이 장에서는 공용 및 개인용 API 엔드포인트 모두에 대한 보안 개선 사항을 권장합니다.

16.1. API 엔드포인트 구성 권장 사항

이 섹션에서는 API 엔드포인트를 강화하기 위한 권장 사항에 대해 설명합니다.

16.1.1. 내부 API 통신

OpenStack은 공용, 내부 관리자 및 개인 API 엔드포인트를 둘 다 제공합니다. 기본적으로 OpenStack 구성 요소는 공개적으로 정의된 엔드포인트를 사용합니다. 적절한 보안 도메인 내에서 API 엔드포인트를 사용하도록 이러한 구성 요소를 구성하는 것이 좋습니다. 내부 관리 엔드포인트에서는 keystone에 대한 추가 액세스를 허용하므로 이를 추가로 격리하는 것이 바람직할 수 있습니다.

서비스는 OpenStack 서비스 카탈로그를 기반으로 각 API 엔드포인트를 선택합니다. 이러한 서비스는 나열된 공용 또는 내부 API 엔드포인트 값을 준수하지 못할 수 있습니다. 이로 인해 내부 관리 트래픽이 외부 API 엔드포인트로 라우팅될 수 있습니다.

16.1.2. ID 서비스 카탈로그에서 내부 URL 구성

ID 서비스 카탈로그는 내부 URL을 인식해야 합니다. 이 기능은 기본적으로 사용되지 않지만 구성을 통해 사용할 수 있습니다. 또한 이 동작이 기본값이 되면 예상 변경 사항과 전달될 수 있어야 합니다.

액세스 수준이 다르면 네트워크 수준에서 구성된 엔드포인트를 격리하는 것이 좋습니다. 관리 엔드포인트는 클라우드 관리자가 액세스하기 위한 것입니다. 내부 또는 공용 엔드포인트에서는 사용할 수 없는 keystone 작업에 대한 높은 액세스 권한을 제공합니다. 내부 엔드포인트는 클라우드 내부(예: OpenStack 서비스)를 사용하기 위한 것이며, 일반적으로 배포 네트워크 외부에서는 액세스할 수 없습니다. 공용 엔드포인트는 TLS를 활성화하고 클라우드 사용자가 작동할 배포 외부에서 액세스할 수 있는 유일한 API 엔드포인트입니다.

엔드포인트의 내부 URL 등록은 director가 자동화합니다. 자세한 내용은 https://github.com/openstack/tripleo-heat-

templates/blob/a7857d6dfcc875eb2bc611dd9334104c18fe8ac6/network/endpoints/build_endpoint_map.py 의 내용을 참조하십시오.

16.1.3. 내부 URL의 애플리케이션 구성

일부 서비스에서 특정 API 끝점을 사용하도록 할 수 있습니다. 따라서 적절한 내부 API 엔드포인트에 액세스하도록 다른 서비스의 API를 연결하는 모든 OpenStack 서비스를 명시적으로 구성해야 합니다.

각 프로젝트에 대상 API 엔드포인트를 정의하는 일관되지 않은 방법이 있을 수 있습니다. 향후 OpenStack 릴리스에서는 ID 서비스 카탈로그를 일관되게 사용하여 이러한 불일치를 해결하려고 합니다.

16.1.4. 붙여넣기 및 미들웨어

OpenStack의 대부분의 API 엔드포인트 및 기타 HTTP 서비스는 Python Paste Deploy 라이브러리를 사용합니다. 보안 관점에서 이 라이브러리를 사용하면 애플리케이션 구성을 통해 요청 필터 파이프라인을 조작할 수 있습니다. 이 체인의 각 요소를 미들웨어라고 합니다. 파이프라인에서 필터 순서를 변경하거나 추가 미들웨어를 추가하면 예기치 않은 보안에 영향을 미칠 수 있습니다.

일반적으로 구현자는 미들웨어를 추가하여 OpenStack의 기본 기능을 확장합니다. 비표준 소프트웨어 구성 요소를 HTTP 요청 파이프라인에 추가하여 발생할 수 있는 노출에 대해 신중하게 고려하십시오.

16.1.5. API 끝점 프로세스 격리 및 정책

API 엔드포인트 프로세스, 특히 공용 보안 도메인에 있는 프로세스를 격리해야 합니다. 배포를 허용하는 경우 격리를 높이기 위해 API 엔드포인트를 별도의 호스트에 배포해야 합니다.

16.1.6. 네임스페이스

Linux는 네임스페이스를 사용하여 프로세스를 독립 도메인에 할당합니다. 본 가이드의 다른 부분에서는 시스템 구분을 자세히 다룹니다.

16.1.7. 네트워크 정책

API 엔드포인트는 일반적으로 여러 보안 영역에 걸쳐 있으므로 API 프로세스 분리에 특히 주의해야 합니다. 예를 들어 네트워크 설계 수준에서는 지정된 시스템에 대한 액세스만 제한할 수 있습니다. 자세한 내용은 보안 영역에 대한 지침을 참조하십시오.

신중하게 모델링하면 네트워크 ACL 및 IDS 기술을 사용하여 네트워크 서비스 간에 명시적인 지점 간

통신을 적용할 수 있습니다. 중요한 교차 도메인 서비스로서 이러한 유형의 명시적 적용은 **OpenStack**의 메시지 대기열 서비스에 잘 작동합니다.

정책을 적용하기 위해 서비스, 호스트 기반 방화벽(예: **iptables**), 로컬 정책(**SELinux**) 및 선택적으로 글로벌 네트워크 정책을 구성할 수 있습니다.

16.1.8. 필수 액세스 제어

시스템의 다른 프로세스와 **API** 엔드포인트 프로세스를 서로 분리해야 합니다. 이러한 프로세스의 구성은 **DAC**(임의적 액세스 제어) 및 **MAC**(강제적 액세스 제어)를 통해 해당 프로세스로 제한되어야 합니다. 이러한 향상된 액세스 제어의 목표는 **API** 엔드포인트 보안 침해를 포함하는 데 도움이 되는 것입니다.

16.1.9. API 끝점 속도 제한

속도 제한은 네트워크 기반 애플리케이션에서 수신한 이벤트 빈도를 제어하는 수단입니다. 강력한 속도 제한이 없으면 애플리케이션이 다양한 서비스 거부 공격에 취약해질 수 있습니다. 이는 특히 비슷한 요청 유형 및 작업의 높은 빈도를 허용하도록 설계된 **API**에 특히 해당합니다.

모든 엔드포인트(특히 공용)에서 물리적 네트워크 설계, 속도 제한 프록시 또는 웹 애플리케이션 방화벽을 사용하는 추가적인 보호 계층을 제공하는 것이 좋습니다.

운영자가 속도 제한 기능을 구성하고 구현할 때 **OpenStack** 클라우드 내에서 사용자 및 서비스의 개별 성능 요구 사항을 신중하게 계획하고 구현하는 것이 중요합니다.

Red Hat OpenStack Platform 배포의 경우 모든 서비스가 부하 분산 프록시 뒤에 배치됩니다.

17장. 페더레이션 구현

주의

Red Hat은 현재 페더레이션을 지원하지 않습니다. 이 기능은 테스트에만 사용해야 하며 프로덕션 환경에 배포해서는 안 됩니다.

17.1. RED HAT SINGLE SIGN-ON을 사용하여 IDM과 통합

Red Hat Single Sign-On(RH-SSO)을 사용하여 **OpenStack** 인증(**authN**)을 위해 **IdM** 사용자를 통합할 수 있습니다. **Federation**을 사용하면 **IdM** 사용자가 **OpenStack** 서비스에 대한 자격 증명을 표시하지 않고도 **OpenStack** 대시보드에 로그인할 수 있습니다. 대신 대시보드에 사용자의 자격 증명이 필요한 경우 사용자를 **RH-SSO(Red Hat Single Sign-On)**로 전달하고 **IdM** 자격 증명을 입력할 수 있습니다. 결과적으로 **RH-SSO**는 사용자가 성공적으로 인증되었음을 대시보드로 돌아간 다음 대시보드를 통해 프로젝트에 액세스할 수 있습니다.

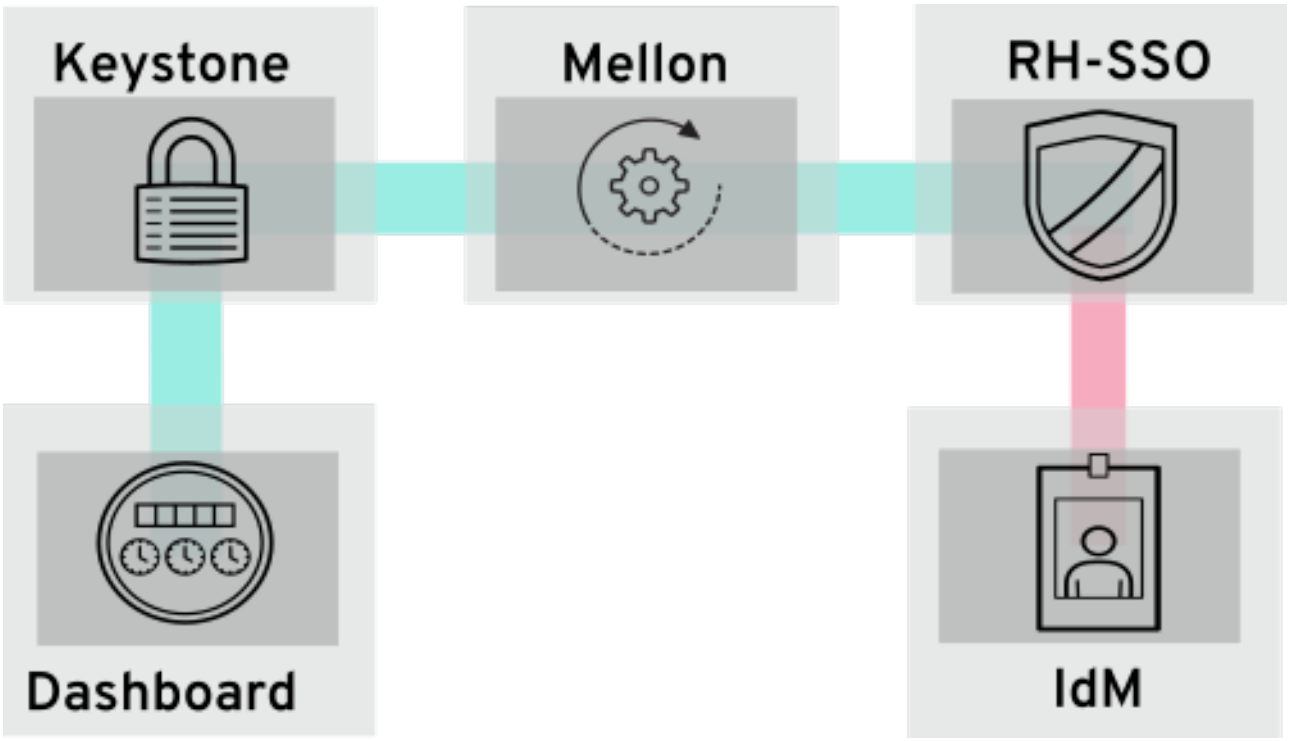
17.2. 페더레이션 워크플로

이 섹션에서는 **keystone, RH-SSO** 및 **IdM**이 서로 상호 작용하는 방법에 대해 설명합니다. **OpenStack**의 페더레이션에서는 **ID** 공급자 및 서비스 공급자의 개념을 사용합니다.

IdM(Identity Provider) - 사용자 계정을 저장하는 서비스입니다. 이 경우 **IdM**에서 보유하는 사용자 계정이 **RH-SSO**를 사용하여 **Keystone**에 제공됩니다.

Service Provider (SP) - IdP의 사용자 인증이 필요한 서비스입니다. 이 경우 **keystone**은 **IdM** 사용자에게 대시보드 액세스 권한을 부여하는 서비스 프로바이더입니다.

아래 다이어그램에서 **keystone(SP)**은 필요한 **SAML2 WebSSO**를 제공하는 **RH-SSO(IdP)**와 통신합니다. **RH-SSO**는 다른 **IdP**의 범용 어댑터도 사용할 수 있습니다. 이 구성에서는 **RH-SSO**의 **keystone**을 가리키고 **RH-SSO**는 지원하는 **ID** 프로바이더(인증 모듈이라고도 함)에 대한 요청을 전달합니다. 여기에는 현재 **IdM** 및 **Active Directory**가 포함됩니다. 이러한 작업은 각 **sysadmin**이 **SP(Service Provider)** 및 **IdM(Identity Provider)** 교환 메타데이터를 보유하여 신뢰하기로 결정합니다. 그 결과 **IdP**가 자신 있게 어설션을 수행할 수 있으며 **SP**는 이러한 어설션을 받을 수 있습니다.



자세한 내용은 페더레이션 가이드 https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html-single/federate_with_identity_service/를 참조하십시오.