Red Hat OpenStack Platform 17.1

强化 Red Hat OpenStack Platform

最佳实践、合规性和安全强化

Last Updated: 2024-05-23

Red Hat OpenStack Platform 17.1 强化 Red Hat OpenStack Platform

最佳实践、合规性和安全强化

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

本指南提供有关强化 Red Hat OpenStack Platform 环境安全性的良好实践建议和概念性信息。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

在 JIRA 中提供文档反馈

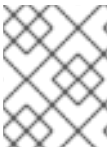
使用 Create Issue 表单对文档提供反馈。JIRA 问题将在 Red Hat OpenStack Platform Jira 项目中创建，您可以在其中跟踪您的反馈进度。

1. 确保您已登录到 JIRA。如果您没有 JIRA 帐户，请创建一个帐户来提交反馈。
2. 点击以下链接打开 **Create Issue** 页面： Create Issue
3. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节或章节号以及问题的详细描述。不要修改表单中的任何其他字段。
4. 点 **Create**。

第 1 章 安全性简介

使用 Red Hat Openstack Platform (RHOSP)提供的工具，在规划和操作中优先选择安全性，以满足用户的隐私性和数据安全性的期望。无法实施安全标准会导致停机或数据泄露。您的用例可能会受到要求通过审计和合规性流程的法律。

- 有关强化 Ceph 的信息，请参阅 数据安全性和强化指南。



注意

按照本指南中的说明强化环境的安全性。但是，这些建议不能保证安全性或合规性。您必须根据环境的唯一要求评估安全性。

1.1. RED HAT OPENSTACK PLATFORM 安全性

默认情况下，Red Hat OpenStack Platform (RHOSP) director 使用以下工具和访问控制来创建 overcloud，以提高安全性：

SElinux

SELinux 通过提供要求每个进程都有每个操作的访问控制来为 RHOSP 提供安全增强。

Podman

Podman 作为容器工具是 RHOSP 的一个安全选项，因为它不使用客户端/服务器模型，它需要具有 root 访问权限的进程才能正常工作。

系统访问限制

您只能使用 director 在 overcloud 部署期间为 tripleo-admin 创建的 SSH 密钥或您在 overcloud 上创建的 SSH 密钥来登录 overcloud 节点。您不能使用带有密码的 SSH 登录 overcloud 节点，或使用 root 登录 overcloud 节点。

您可以根据机构的需要和信任级别，使用以下额外安全功能配置 director：

- 公共 TLS 和 TLS-everywhere
- 硬件安全模块与 OpenStack Key Manager (barbican)集成
- 签名的镜像和加密卷
- 使用工作流执行对密码和 fernet 密钥进行轮转

1.2. 了解 RED HAT OPENSTACK PLATFORM ADMIN 角色

当您为用户分配 **admin** 角色时，此用户具有查看、更改、创建或删除任何项目的任何资源的权限。此用户可以创建可在项目间访问的资源，如公开可用的 glance 镜像或提供商网络。此外，具有 **admin** 角色的用户可以创建或删除用户并管理角色。

为您为用户分配 **admin** 角色的项目是执行 **openstack** 命令的默认项目。例如，如果一个 **admin** 用户在一个名为 **development** 的项目中运行以下命令，将会在名为 **development** 项目中创建一个名为 **internal-network** 的网络。

```
openstack network create internal-network
```

admin 用户可以使用 **--project** 参数在任何项目中创建 **internal-network** ：

```
openstack network create internal-network --project testing
```

1.3. 识别 RED HAT OPENSTACK PLATFORM 中的安全区

安全区包括共享共同安全问题的资源、应用程序、网络和服务器。设计安全区，以便具有共同的身份验证和授权要求和用户。您可以根据您的云架构、环境中可接受的信任程度以及您组织的标准化要求，将自己的安全区定义为需要精细的安全区。区域及其信任要求可能会因云实例是公共、私有还是混合而不同。

例如，您可以将 Red Hat OpenStack Platform 的默认安装划分为以下区：

表 1.1. 安全区

zone	网络	详情
公开	external	public 区域托管外部网络、公共 API 和浮动 IP 地址，以用于实例的外部连接。此区域允许从您的管理控制之外的网络访问，并且是云基础架构的不受信任的区域。
Guest	tenant	guest 区域托管项目网络。对于允许不受限制访问实例的公共和私有云供应商，它不被信任。
存储访问	storage, storage_mgmt	存储访问区域用于存储管理、监控和集群和存储流量。
控制	ctlplane, internal_api, ipmi	control 区域还包括 undercloud、主机操作系统、服务器硬件、物理网络和 Red Hat OpenStack Platform director 控制平面。

1.4. 在 RED HAT OPENSTACK PLATFORM 中查找安全区

运行以下命令来收集有关 Red Hat OpenStack Platform 部署的物理配置的信息：

先决条件

- 已安装 Red Hat OpenStack Platform 环境。
- 以 stack 身份登录 director。

流程

1. 源 **stackrc** ：

```
$ source /home/stack/stackrc
```

2. 运行 **openstack subnet list**，将分配的 ip 网络与关联的区匹配：

```
openstack subnet list -c Name -c Subnet
```

```
+--------------------+-----------------+
| Name               | Subnet          |
+--------------------+-----------------+
| ctlplane-subnet    | 192.168.101.0/24 |
| storage_mgmt_subnet | 172.16.105.0/24  |
| tenant_subnet      | 172.16.102.0/24  |
| external_subnet    | 10.94.81.0/24    |
| internal_api_subnet | 172.16.103.0/24  |
| storage_subnet     | 172.16.104.0/24  |
+--------------------+-----------------+
```

3. 运行 **openstack server list** 以列出您基础架构中的物理服务器：

```
openstack server list -c Name -c Networks
+------------------------+------------------------+
| Name                   | Networks               |
+------------------------+------------------------+
| overcloud-controller-0  | ctlplane=192.168.101.15 |
| overcloud-controller-1  | ctlplane=192.168.101.19 |
| overcloud-controller-2  | ctlplane=192.168.101.14 |
| overcloud-novacompute-0 | ctlplane=192.168.101.18 |
| overcloud-novacompute-2 | ctlplane=192.168.101.17 |
| overcloud-novacompute-1 | ctlplane=192.168.101.11 |
+------------------------+------------------------+
```

4. 使用 **openstack server list** 命令中的 **ctlplane** 地址来查询物理节点的配置：

```
ssh tripleo-admin@192.168.101.15 ip addr
```

1.5. 连接安全区

您必须仔细配置跨越多个安全区（具有不同信任级别或身份验证要求）的任何组件。这些连接通常是网络架构中弱点。确保配置这些连接以满足所连接任何区域的最高信任级别的安全要求。在很多情况下，连接的区的安全控制是主要问题，因为攻击的可能性较高。区域满足额外的潜在攻击点，并为攻击者增加将攻击迁移到更敏感的部署部分的机会。

在某些情况下，OpenStack 操作员可能考虑使用比它所驻留的任何区域更高的标准来保护集成点。根据上述 API 端点示例，如果这些区域没有完全隔离，则可能会以公共区的公共 API 端点为目标，利用这个范围。

OpenStack 的设计使得分离安全区比较困难。由于核心服务通常至少跨越两个区域，因此在将安全控制应用到它们时，必须考虑特殊考虑。

1.6. 威胁缓解

大多数类型的云部署、公共、私有或混合都暴露在某种形式的安全威胁中。以下实践有助于缓解安全威胁：

- 应用最小特权的原则。
- 对内部和外部接口使用加密。
- 使用集中式身份管理。

- 使 Red Hat OpenStack Platform 保持更新。

计算服务可以为恶意攻击者提供 DDoS 和 brute 强制攻击的工具。防止方法包括出口安全组、流量检查、入侵检测系统和客户教育和认知。对于公共网络访问或可访问公共网络（如互联网）的部署，请确保进程和基础架构就位，以检测和解决出站滥用。

其他资源

- 使用 Ansible 实施 TLS-e
- 将 OpenStack 身份(keystone)与红帽身份管理器(IdM)集成
- 执行 Red Hat OpenStack Platform 的次要更新

第 2 章 安全增强

以下小节提供了强化 overcloud 安全性的一些建议。

2.1. 使用安全 ROOT 用户访问

overcloud 镜像自动包含 **root** 用户的强化安全性。例如，每个部署的 overcloud 节点都自动禁用对 **root** 用户的直接 SSH 访问。您仍然可以访问 overcloud 节点上的 **root** 用户。每个 overcloud 节点都有一个 **tripleo-admin** 用户帐户。此用户帐户包含 undercloud 公共 SSH 密钥，它提供 SSH 访问，无需从 undercloud 到 overcloud 节点的密码。

先决条件

- 已安装 Red Hat OpenStack Platform director 环境。
- 以 stack 身份登录 director。

流程

1. 在 undercloud 节点上，以 **tripleo-admin** 用户身份通过 SSH 登录 overcloud 节点。
2. 通过 **sudo -i** 切换到 **root** 用户。

2.2. 将服务添加到 OVERCLOUD 防火墙

部署 Red Hat OpenStack Platform 时，每个核心服务都会在每个 overcloud 节点上使用一组默认的防火墙规则进行部署。您可以使用 **ExtraFirewallRules** 参数为其他服务创建规则来打开端口，或者创建规则来限制服务。

每个规则名称成为对应 **iptables** 规则的注释。每个规则名称都以三位前缀开头，以帮助 Puppet 订购最终 **iptables** 文件中的规则。默认的 Red Hat OpenStack Platform 规则使用 000 到 200 范围中的前缀。为新服务创建规则时，请为名称添加前缀，其三位数字高于 200。

流程

1. 使用字符串在 **ExtraFireWallRules** 参数下定义每个规则名称。您可以在规则名称下使用以下参数来定义规则：
   - dport:: 与规则关联的目的地端口。
   - proto:: 与规则关联的协议。默认为 **tcp**。
   - action:: 与规则关联的操作策略。默认为 **接受**。
   - source:: 与规则关联的源 IP 地址。
   以下示例演示了如何使用规则为自定义应用程序打开附加端口：

```
cat > ~/templates/firewall.yaml <<EOF
parameter_defaults:
  ExtraFirewallRules:
    '300 allow custom application 1':
      dport: 999
      proto: udp
    '301 allow custom application 2':
```

```
    dport: 8081
    proto: tcp
EOF
```

注意

如果没有设置 **action** 参数，则结果将 **接受**。您只能将 **action** 参数设置为 **drop**、**insert** 或 **append**。

2. 在 **openstack overcloud deloy** 命令中包含 **~/templates/firewall.yaml** 文件。包括部署所需的所有模板：

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

2.3. 从 OVERCLOUD 防火墙中删除服务

您可以使用规则来限制服务。您在规则名称中使用的数字决定了规则将插入的 **iptables** 中的位置。以下流程演示了如何将 **rabbitmq** 服务限制为 InternalAPI 网络。

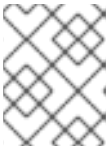
流程

1. 在 Controller 节点上，查找 **rabbitmq** 的默认 **iptables** 规则数：

```
[tripleo-admin@overcloud-controller-2 ~]$ sudo iptables -L | grep rabbitmq
ACCEPT     tcp  --  anywhere             anywhere             multiport dports vtr-
emulator,epmd,amqp,25672,25673:25683 state NEW /* 109 rabbitmq-bundle ipv4 */
```

2. 在环境文件 uder **parameter_defaults** 中，使用 **ExtraFirewallRules** 参数将 **rabbitmq** 限制到 InternalApi 网络。该规则的编号低于默认的 **rabbitmq** 规则编号或 109 ：

```
cat > ~/templates/firewall.yaml <<EOF
parameter_defaults:
  ExtraFirewallRules:
    '098 allow rabbit from internalapi network':
      dport:
      - 4369
      - 5672
      - 25672
      proto: tcp
      source: 10.0.0.0/24
    '099 drop other rabbit access':
      dport:
      - 4369
      - 5672
      - 25672
      proto: tcp
      action: drop
EOF
```



注意

如果没有设置 **action** 参数，则结果将 **接受**。您只能将 **action** 参数设置为 **drop**、**insert** 或 **append**。

3. 在 **openstack overcloud deloy** 命令中包含 **~/templates/firewall.yaml** 文件。包括部署所需的所有模板：

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

2.4. 更改简单网络管理协议(SNMP)字符串

director 为您的 overcloud 提供默认的只读 SNMP 配置。建议更改 SNMP 字符串，以减少未授权用户了解您的网络设备的风险。

注意

当使用字符串参数配置 **ExtraConfig** 接口时，您必须使用以下语法来确保 heat 和 Hiera 不将字符串解释为布尔值： **"'<VALUE>'"**。

使用 overcloud 的环境文件中的 **ExtraConfig** hook 设置以下 hieradata：

SNMP 传统访问控制设置

snmp::ro_community

IPv4 只读 SNMP 社区字符串.默认值为 **public**。

snmp::ro_community6

IPv6 只读 SNMP 社区字符串.默认值为 **public**。

snmp::ro_network

允许 **RO 查询** 守护进程的网络。这个值可以是字符串或数组。默认值为 **127.0.0.1**。

snmp::ro_network6

允许使用 IPv **6** 查询 守护进程的网络。这个值可以是字符串或数组。默认值为 **::1/128**。

tripleo::profile::base::snmp::snmpd_config

要作为安全 valve 添加到 ***snmpd.conf*** 文件中的行数组。默认值为 **[]**。有关所有可用选项，请参阅 SNMP 配置文件 网页。

例如：

```
parameter_defaults:
  ExtraConfig:
    snmp::ro_community: mysecurestring
    snmp::ro_community6: myv6securestring
```

这会更改所有节点上的只读 SNMP 社区字符串。

SNMP 基于视图的访问控制设置(VACM)

snmp::com2sec

VACM com2sec 映射的数组。必须提供 SECNAME、SOURCE 和 COMMUNITY。

snmp::com2sec6

VACM com2sec6 映射的数组。必须提供 SECNAME、SOURCE 和 COMMUNITY。

例如：

```
parameter_defaults:
  ExtraConfig:
    snmp::com2sec: ["notConfigUser default mysecurestring"]
    snmp::com2sec6: ["notConfigUser default myv6securestring"]
```

这会更改所有节点上的只读 SNMP 社区字符串。

如需更多信息，请参阅 **snmpd.conf** 手册页。

2.5. 使用 OPEN VSWITCH 防火墙

您可以将安全组配置为使用 Red Hat OpenStack Platform director 中的 Open vSwitch (OVS)防火墙驱动程序。使用 **NeutronOVSFirewallDriver** 参数指定您要使用的防火墙驱动程序：

- **iptables_hybrid** - 配置网络服务(neutron)以使用基于 iptables/hybrid 的实现。
- **openvswitch** - 将网络服务配置为使用基于 OVS 防火墙流的驱动程序。

openvswitch 防火墙驱动程序包括更高的性能，并减少用于将客户机连接到项目网络的接口和网桥的数量。

重要

多播流量由 Open vSwitch (OVS)防火墙驱动程序与 iptables 防火墙驱动程序不同。使用 iptables 时，默认情况下 VRRP 流量会被拒绝，您必须在安全组规则中为任何 VRRP 流量启用 VRRP 才能访问端点。使用 OVS 时，所有端口共享相同的 OpenFlow 上下文，并且每个端口无法单独处理多播流量。由于安全组不适用于所有端口（例如，路由器上的端口），因此 OVS 使用 **NORMAL** 操作并将多播流量转发到 RFC 4541 指定的所有端口。

注意

iptables_hybrid 选项与 OVS-DPDK 不兼容。 **openvswitch** 选项与 OVS Hardware Offload 不兼容。

在 **network-environment.yaml** 文件中配置 **NeutronOVSFirewallDriver** 参数：

```
NeutronOVSFirewallDriver: openvswitch
```

- **NeutronOVSFirewallDriver** ：配置在实施安全组时要使用的防火墙驱动程序名称。可能的值取决于您的系统配置。某些示例为 **noop**,**openvswitch**, 和 **iptables_hybrid**。空字符串的默认值会产生支持的配置。

第 3 章 记录您的 RHOSP 环境

记录系统组件、网络、服务和软件对于识别安全问题、攻击向量和可能的安全区桥接点非常重要。Red Hat OpenStack Platform (RHOSP)部署的文档应包含以下信息：

- RHOSP 产品、开发和测试环境中的系统组件、网络、服务和软件的描述。
- 任何临时资源（如虚拟机或虚拟磁盘卷）的清单。

3.1. 记录系统角色

Red Hat OpenStack Platform (RHOSP)部署中的每个节点都提供特定的角色，可以贡献云的基础架构或提供云资源。

贡献基础架构的节点运行与云相关的服务，如消息排队服务、存储管理、监控、网络以及支持云操作和调配所需的其他服务。基础架构角色示例包括：

- Controller
- Networker
- 数据库
- Telemetry

提供云资源的节点为云上运行的实例提供计算或存储容量。资源角色示例包括：

- CephStorage
- Compute
- ComputeOvsDpdk
- ObjectStorage

记录下您的环境中使用的系统角色。这些角色可以在用于部署 RHOSP 的模板中标识。例如，环境中每个角色都有一个 NIC 配置文件。

流程

1. 检查您的部署的现有模板，以了解用于指定当前正在使用的角色的文件。在您的环境中每个角色都有一个 NIC 配置文件。在以下示例中，RHOSP 环境包括 **ComputeHCI** 角色、**Compute** 角色和 **Controller** 角色：

```
$ cd ~/templates
$ tree
.
├── environments
│   └── network-environment.yaml
├── hci.yaml
├── network
│   └── config
│       └── multiple-nics
│           ├── computehci.yaml
│           ├── compute.yaml
│           └── controller.yaml
```

```
├── network_data.yaml
├── plan-environment.yaml
└── roles_data_hci.yaml
```

2. RHOSP 环境的每个角色都执行许多相关的服务。您可以通过检查 **roles** 文件来记录各个角色使用的服务。

   a. 如果为您的模板生成了一个 **roles** 文件，您可以在 **~/templates** 目录中找到该文件：

   ```
   $ cd ~/templates
   $ find . -name *role*
   > ./templates/roles_data_hci.yaml
   ```

   b. 如果没有为您的模板生成 **roles** 文件，您可以为当前用于检查文档的角色生成一个角色：

   ```
   $ openstack overcloud roles generate \
   > --roles-path /usr/share/openstack-tripleo-heat-templates/roles \
   > -o roles_data.yaml Controller Compute
   ```

3.2. 创建硬件清单

您可以通过查看内省期间收集的数据来检索您的 Red Hat OpenStack Platform 部署硬件信息。内省从有关 CPU、内存、磁盘等的节点收集硬件信息。

先决条件

- 已安装 Red Hat OpenStack Platform director 环境。
- 您已内省了用于 Red Hat OpenStack Platform 部署的节点。
- 以 stack 身份登录 director。

流程

1. 在 undercloud 中，提供 **stackrc** 文件：

   ```
   $ source ~/stackrc
   ```

2. 列出环境中的节点：

   ```
   $ openstack baremetal node list -c Name
   +--------------+
   | Name         |
   +--------------+
   | controller-0 |
   | controller-1 |
   | controller-2 |
   | compute-0    |
   | compute-1    |
   | compute-2    |
   +--------------+
   ```

3. 对于每个收集信息的裸机节点，并运行以下命令来检索内省数据：

```
$ openstack baremetal introspection data save <node> | jq
```

将 **<node** > 替换为在第 1 步中获得的列表中的节点名称。

4. 可选： 要将输出限制为特定类型的硬件，您可以检索清单密钥列表并查看特定键的内省数据：

   a. 运行以下命令，从内省数据中获取顶层键列表：

```
$ openstack baremetal introspection data save controller-0 | jq '.inventory | keys'

[
  "bmc_address",
  "bmc_v6address",
  "boot",
  "cpu",
  "disks",
  "hostname",
  "interfaces",
  "memory",
  "system_vendor"
]
```

   b. 选择一个键，如 **disks**，并运行以下命令以获取更多信息：

```
$ openstack baremetal introspection data save controller-1 | jq '.inventory.disks'
[
  {
    "name": "/dev/sda",
    "model": "QEMU HARDDISK",
    "size": 85899345920,
    "rotational": true,
    "wwn": null,
    "serial": "QM00001",
    "vendor": "ATA",
    "wwn_with_extension": null,
    "wwn_vendor_extension": null,
    "hctl": "0:0:0:0",
    "by_path": "/dev/disk/by-path/pci-0000:00:01.1-ata-1"
  }
]
```

3.3. 创建软件清单

记录下在 Red Hat OpenStack Platform (RHOSP)基础架构中部署的节点上使用的软件组件。系统数据库、RHOSP 软件服务和支持组件（如负载均衡器、DNS 或 DHCP 服务）在评估库、应用程序或软件类方面的影响至关重要。

- 已安装 Red Hat OpenStack Platform 环境。
- 以 stack 身份登录 director。

流程

1. 确保您知道可能受到恶意活动的系统和服务的入口点。在 undercloud 上运行以下命令：

```
$ cat /etc/hosts
$ source stackrc ; openstack endpoint list
$ source overcloudrc ; openstack endpoint list
```

2. RHOSP 部署到容器化服务中，因此您可以通过检查在该节点上运行的容器来查看 overcloud 节点上的软件组件。使用 **ssh** 连接到 overcloud 节点并列出正在运行的容器。例如，若要查看 **compute-0** 上的 overcloud 服务，请运行以下命令：

```
$ ssh tripleo-admin@compute-0 podman ps
```

第 4 章 身份和访问管理

Identity 服务(keystone)为 Red Hat OpenStack Platform 环境中的云用户提供身份验证和授权。您可以使用身份服务直接进行最终用户身份验证，或者将其配置为使用外部身份验证方法来满足安全要求或与您当前的身份验证基础架构匹配。

4.1. RED HAT OPENSTACK PLATFORM FERNET 令牌

Fernet 是替换 **UUID** 令牌提供程序的默认令牌提供程序。默认情况下，每个 fernet 令牌保持在一小时内有效。这允许用户执行一系列任务，而无需重新进行身份验证。

身份验证后，身份服务(keystone)：

- 发出加密的 bearer 令牌，称为 fernet 令牌。此令牌代表您的身份。
- 授权您基于您的角色执行操作。

其他资源

- 在 overcloud 中使用 Fernet 密钥进行加密

4.2. OPENSTACK IDENTITY 服务实体

Red Hat OpenStack Identity 服务 (keystone) 可以识别以下实体：

用户

OpenStack Identity 服务(keystone)用户是身份验证的原子单元。必须为用户分配项目的角色才能进行身份验证。

组

OpenStack Identity 服务组是用户的逻辑分组。可以为组提供特定角色下项目的访问权限。管理组而不是用户可简化角色的管理。

角色

OpenStack Identity 服务角色定义了 OpenStack API，可供分配了这些角色的用户或组访问。

项目

OpenStack Identity 服务项目是隔离的用户组，其对物理资源共享配额具有共同访问权限，以及从这些物理资源构建的虚拟基础架构。

Domains

OpenStack Identity 服务域是项目、用户和组的高级安全界限。您可以使用 OpenStack 身份域来集中管理所有基于 keystone 的身份组件。Red Hat OpenStack Platform 支持多个域。您可以使用单独的身份验证后端代表不同域的用户。

4.3. 使用 KEYSTONE 进行身份验证

您可以调整 OpenStack Identity 服务(keystone)所需的身份验证安全要求。

表 4.1. Identity 服务身份验证参数

参数	描述

KeystoneLockoutDuration	超过用户帐户的失败尝试次数上限（如 **KeystoneLockoutFailureAttempts**指定）的最大数量被锁定。
KeystoneLockoutFailureAttempts	在 **KeystoneLockoutDuration** 指定的秒数内，用户可以在用户帐户锁定前验证的次数上限。
KeystoneMinimumPasswordAge	在用户可以更改密码之前必须使用密码的天数。这可防止用户立即更改密码，以擦除密码历史记录并重复使用旧密码。
KeystoneUniqueLastPasswordCount	这将控制在历史记录中保留的之前用户密码迭代的数量，以便强制新创建的密码是唯一的。

其他资源

- 身份(keystone)参数。

4.4. 使用 IDENTITY 服务 HEAT 参数停止无效的登录尝试

重复失败的登录尝试可能是试图的攻击。您可以使用 Identity Service 在重复失败的登录尝试后限制对帐户的访问。

先决条件

- 已安装 Red Hat OpenStack Platform director 环境。
- 以 stack 身份登录 director。

流程

1. 要配置用户在用户帐户锁定前无法进行身份验证的最大次数，请在环境文件中设置 **KeystoneLockoutFailureAttempts** 和 **KeystoneLockoutDuration** heat 参数的值。在以下示例中，**KeystoneLockoutDuration** 设置为一小时：

```
parameter_defaults
  KeystoneLockoutDuration: 3600
  KeystoneLockoutFailureAttempts: 3
```

2. 在部署脚本中包含环境文件。当您在之前部署的环境中运行部署脚本时，会使用附加参数进行更新：

```
openstack overcloud deploy --templates \
...
-e keystone_config.yaml
...
```

4.5. 使用外部身份提供程序进行身份验证

您可以使用外部身份提供程序(IdP)向 OpenStack 服务提供程序(SP)进行身份验证。SPS 是由 OpenStack 云提供的服务。

当您使用单独的 IdP 时，外部验证凭证会独立于其他 OpenStack 服务使用的数据库。这种分离降低了存储凭证的风险。

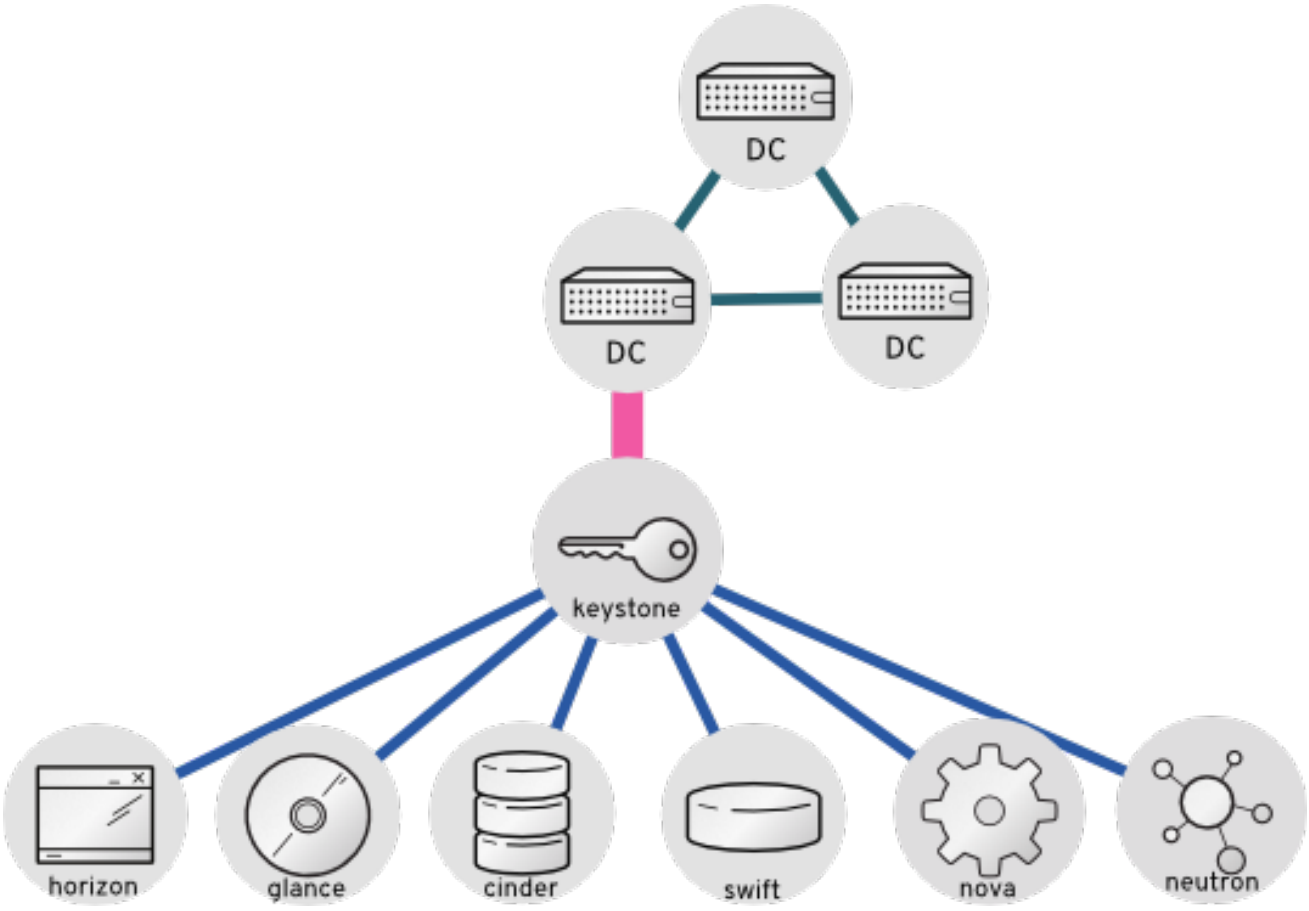
每个外部 IdP 都有一个到 OpenStack Identity 服务(keystone)域的一对一映射。您可以在 Red Hat OpenStack Platform 中有多个已存在的域。

外部身份验证提供了一种方式，可以在不创建额外的身份的情况下使用现有凭证访问 Red Hat OpenStack Platform 中的资源。凭证由用户的 IdP 维护。

您可以使用 Red Hat Identity Management (IdM)和 Microsoft Active Directory 域服务(AD DS)等 IdP 进行身份管理。在此配置中，OpenStack Identity 服务对 LDAP 用户数据库具有只读访问权限。基于用户或组角色的 API 访问管理由 keystone 执行。使用 OpenStack Identity 服务将角色分配给 LDAP 帐户。

4.5.1. LDAP 集成如何工作

在下图中，keystone 使用加密的 LDAPS 连接来连接到 Active Directory 域控制器。当用户登录到 horizon 时，keystone 会收到提供的用户凭证并将其传递给 Active Directory。



其他资源

- 将 OpenStack Identity (keystone)与 Active Directory 集成
- 将 OpenStack 身份(keystone)与红帽身份管理器(IdM)集成
- 将 director 配置为使用域特定的 LDAP 后端

第 5 章 使用 TLS 和 PKI 保护 RED HAT OPENSTACK 部署

Red Hat OpenStack Platform 由多个网络和端点组成，它们处理您可以保护的敏感或机密数据。使用传输层安全(TLS)时，您可以使用对称密钥加密保护流量。密钥和密码在 TLS 握手中协商，这需要在称为证书颁发机构(CA)的中间人中通过共享的信任来验证服务器的身份。

公钥基础架构(PKI)是通过证书颁发机构验证实体的框架。

5.1. 公钥基础架构组件(PKI)

PKI 的核心组件在下表中显示：

表 5.1. 主要术语

术语	定义
结束实体	通过数字证书验证其自身的用户、进程或系统。
证书颁发机构(CA)	CA 是一个受端点实体信任的实体，以及验证最终实体的依赖方。
依赖方	依赖方接收作为终端实体验证的数字证书，并具有验证数字证书的能力。
数字证书	签名的公钥证书有一个可验证的实体和公钥，并由 CA 发布。当 CA 为证书签名时，它会从使用其私钥加密的证书创建消息摘要。您可以使用与 CA 关联的公钥验证签名。X.509 标准用于定义证书。
注册机构(RA)	RA 是一个可选的专用授权，可以在 CA 发布证书之前执行管理功能，如身份验证结束实体。如果没有 RA，CA 会验证结束实体。
证书撤销列表(CRL)	CRL 是已撤销的证书序列号列表。带有撤销序列号的证书的最终实体在 PKI 模型中不被信任。
CRL 签发者	CA 将发布证书撤销列表的可选系统。
证书存储库	端点实体证书和证书撤销列表的位置会被存储和查询。

5.2. 证书颁发机构要求和建议

您必须获得由广泛认可的证书认证机构(CA)签名的证书，以便公开可用的 Red Hat OpenStack Platform Dashboards 或公开可访问的 API。

您必须为每个使用 TLS 保护的端点提供 DNS 域或子域。您提供的域用于创建 CA 发布的证书。客户使用 DNS 名称访问仪表板或 API，以便 CA 可以验证端点。

红帽建议使用单独的和内部管理的 CA 来保护内部流量。这使得云部署器能够保持对其私钥基础架构(PKI)实施的控制，并方便请求、签名和部署内部系统证书。

您可以在 overcloud 端点上启用 SSL/TLS。由于在任何位置配置 TLS (TLS-e)所需的证书数量，director 可与红帽身份管理(IdM)服务器集成，以充当证书颁发机构并管理 overcloud 证书。有关配置 TLS-e 的更多信息，请参阅使用 Ansible 实施 TLS-e。

要检查 OpenStack 组件中的 TLS 支持状态，请参阅 TLS 启用状态列表。

如果要将 SSL 证书与您自己的证书颁发机构搭配使用，请参阅在 overcloud 公共端点 上启用 SSL/TLS。

注意

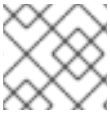
这将只在公开访问的端点中使用 SSL/TLS 配置 Red Hat OpenStack Platform。

5.3. 识别环境中的 TLS 版本

重要

Red Hat OpenStack 平台已弃用 TLS 版本 1.0。另外，对于 NIST 批准，必须至少使用 TLS 1.2。如需更多信息，请参阅 选择、配置和使用传输层安全(TLS)实施的指南。

您可以使用 **cipherscan** 来决定部署提供的 TLS 版本。Cipherscan 可以从 https://github.com/mozilla/cipherscan 克隆。这个示例输出演示了从 **horizon** 收到的结果：



注意

从非生产环境系统运行 **cipherscan**，因为它可能会在首次运行时安装其他依赖项。

流程

- 针对 Dashboard 服务的可访问的 **URL** 运行密码扫描：

```
$ ./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
```

扫描服务器时，Cipherscan 会公告对特定 TLS 版本的支持，这是它将被协商的最高 TLS 版本。如果目标服务器正确遵循 TLS 协议，它将响应相互支持的最高版本，这可能低于 Cipherscan 最初公告的内容。如果服务器使用该特定版本与客户端建立连接，则它不被视为接受该协议版本。如果没有建立连接（使用指定的版本或任何较低版本），则对该版本的协议具有容错被视为存在。例如：

```
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
```

在这个输出中，**TLS 1.0** 和 **TLS 1.1** 的不容限被报告为 **PRESENT**，这意味着无法建立连接，并且 Cipherscan 在广告支持时无法连接。因此，在扫描的服务器上不启用该协议的这些（及任何低）版本是合理的。

5.4. OPENSTACK 的身份管理(IDM)服务器建议

红帽提供了以下信息，以帮助您集成 IdM 服务器和 OpenStack 环境。

有关为 IdM 安装准备 Red Hat Enterprise Linux 的详情，请参考 安装身份管理。

运行 **ipa-server-install** 命令来安装和配置 IdM。您可以使用命令参数跳过交互式提示。使用以下建议，以便您的 IdM 服务器可以与 Red Hat OpenStack Platform 环境集成：

表 5.2. 参数建议

选项	建议
--admin-password	请注意您提供的值。在配置 Red Hat OpenStack Platform 以使用 IdM 时，您将需要此密码。

选项	建议
--ip-address	请注意您提供的值。undercloud 和 overcloud 节点需要网络访问此 IP 地址。
--setup-dns	使用这个选项在 IdM 服务器上安装集成的 DNS 服务。undercloud 和 overcloud 节点使用 IdM 服务器进行域名解析。
--auto-forwarders	使用这个选项将 **/etc/resolv.conf** 中的地址用作 DNS 转发器。
--auto-reverse	使用这个选项解析 IdM 服务器 IP 地址的反向记录和区域。如果无法解析反向记录或区域，IdM 会创建反向区域。这简化了 IdM 部署。
--ntp-server, **--ntp-pool**	您可以使用这两个选项或其中任何一个选项来配置 NTP 源。IdM 服务器和 OpenStack 环境必须具有正确的和同步时间。

您必须打开 IdM 所需的防火墙端口，以启用与 Red Hat OpenStack Platform 节点的通信。如需更多信息，请参阅打开 IdM 所需的端口。

其他资源

- 配置和管理身份管理
- Red Hat Identity Management 文档

5.5. 使用 ANSIBLE 实施 TLS-E

您可以使用新的 **tripleo-ipa** 方法在 overcloud 端点上启用 SSL/TLS，称为 TLS 随处(TLS-e)。由于所需的证书数量，Red Hat OpenStack Platform 与 Red Hat Identity Management (IdM)集成。当您使用 **tripleo-ipa** 配置 TLS-e 时，IdM 是证书颁发机构。

先决条件

- 确保完成 undercloud 的所有配置步骤，如创建 stack 用户。如需了解更多详细信息，请参阅使用 director 安装和管理 Red Hat OpenStack Platform。
- DNS 服务器的 IP 地址是在 undercloud 上配置为 IdM 服务器的 IP 地址。以下参数之一必须在 **undercloud.conf** 文件中配置：
  - **DEFAULT/undercloud_nameservers**
  - **%SUBNET_SECTION%/dns_nameservers**

流程

使用以下步骤在新的 Red Hat OpenStack Platform 安装或您要使用 TLS-e 配置的现有部署上实现 TLS-e。如果在预置备节点上部署带有 TLS-e 的 Red Hat OpenStack Platform，则必须使用此方法。

注意

如果您要为现有环境实施 TLS-e，则需要运行 **openstack undercloud install**、和 **openstack overcloud deploy** 等命令。这些流程是幂等的，仅调整现有的部署配置以匹配更新的模板和配置文件。

1. 配置 **/etc/resolv.conf** 文件：
   在 **/etc/resolv.conf** 中设置 undercloud 上的适当搜索域和名称服务器。例如，如果部署域为 **example.com**，并且 FreeIPA 服务器的域是 **bigcorp.com**，则将以下行添加到 /etc/resolv.conf 中：

```
search example.com bigcorp.com
nameserver $IDM_SERVER_IP_ADDR
```

2. 安装所需的软件：

```
sudo dnf install -y python3-ipalib python3-ipaclient krb5-devel
```

3. 使用特定于您的环境的值导出环境变量：

```
export IPA_DOMAIN=bigcorp.com
export IPA_REALM=BIGCORP.COM
export IPA_ADMIN_USER=$IPA_USER (1)
export IPA_ADMIN_PASSWORD=$IPA_PASSWORD (2)
export IPA_SERVER_HOSTNAME=ipa.bigcorp.com
export UNDERCLOUD_FQDN=undercloud.example.com (3)
export USER=stack
export CLOUD_DOMAIN=example.com
```

(1) (2) IdM 用户凭证是一个管理用户，它可以添加新主机和服务。

(3) **UNDERCLOUD_FQDN** 参数的值与 **/etc/hosts** 中的第一个主机名到 IP 地址映射匹配。

4. 在 undercloud 上运行 **undercloud-ipa-install.yaml** ansible playbook：

```
ansible-playbook \
--ssh-extra-args "-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null" \
/usr/share/ansible/tripleo-playbooks/undercloud-ipa-install.yaml
```

5. 在 undercloud.conf 中添加以下参数

```
undercloud_nameservers = $IDM_SERVER_IP_ADDR
overcloud_domain_name = example.com
```

6. [可选] 如果您的 IPA 域与您的 IPA 域不匹配，请设置 **certmonger_krb_realm** 参数的值：

   a. 在 **/home/stack/hiera_override.yaml** 中设置 **certmonger_krb_realm** 的值：

```
parameter_defaults:
  certmonger_krb_realm = EXAMPLE.COMPANY.COM
```

b. 将 **undercloud.conf** 中的 **custom_env_files** 参数的值设置为 **/home/stack/hiera_override.yaml** ：

```
custom_env_files = /home/stack/hiera_override.yaml
```

7. 部署 undercloud：

```
openstack undercloud install
```

验证

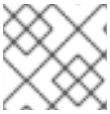
通过完成以下步骤验证 undercloud 是否已正确注册：

1. 列出 IdM 中的主机：

```
$ kinit admin
$ ipa host-find
```

2. 确认 undercloud 上存在 **/etc/novajoin/krb5.keytab**。

```
ls /etc/novajoin/krb5.keytab
```



注意

novajoin 目录名称仅用于传统的命名目的。

在 overcloud 上配置 TLS-e

当您随处部署带有 TLS (TLS-e)的 overcloud 时，Undercloud 和 Overcloud 中的 IP 地址将自动注册到 IdM。

1. 在部署 overcloud 之前，先创建一个 YAML 文件 **tls-parameters.yaml**，其内容类似于以下内容：您选择的值将特定于您的环境：

```
parameter_defaults:
  DnsSearchDomains: ["example.com"]
  CloudDomain: example.com
  CloudName: overcloud.example.com
  CloudNameInternal: overcloud.internalapi.example.com
  CloudNameStorage: overcloud.storage.example.com
  CloudNameStorageManagement: overcloud.storagemgmt.example.com
  CloudNameCtlplane: overcloud.ctlplane.example.com
  IdMServer: freeipa-0.redhat.local
  IdMDomain: redhat.local
  IdMInstallClientPackages: False

resource_registry:
    OS::TripleO::Services::IpaClient: /usr/share/openstack-tripleo-heat-
templates/deployment/ipa/ipaservices-baremetal-ansible.yaml
```

- **OS::TripleO::Services::IpaClient** 参数显示的值会覆盖 **enable-internal-tls.yaml** 文件中的默认设置。您必须确保 **tls-parameters.yaml** 文件遵循 **openstack overcloud deploy** 命令中的 **enable-internal-tls.yaml**。

- 有关用于实现 TLS-e 的参数的更多信息，请参阅 tripleo-ipa 的参数。

2. [可选] 如果您的 IPA 域与您的 IPA 域不匹配，还必须在 **tls-parameters.yaml** 文件中包含 **CertmongerKerberosRealm** 参数的值：

```
CertmongerKerberosRealm: EXAMPLE.COMPANY.COM
```

3. 部署 overcloud。您需要在部署命令中包含 tls-parameters.yaml ：

```
DEFAULT_TEMPLATES=/usr/share/openstack-tripleo-heat-templates/
CUSTOM_TEMPLATES=/home/stack/templates

openstack overcloud deploy \
-e ${DEFAULT_TEMPLATES}/environments/ssl/tls-everywhere-endpoints-dns.yaml \
-e ${DEFAULT_TEMPLATES}/environments/services/haproxy-public-tls-certmonger.yaml \
-e ${DEFAULT_TEMPLATES}/environments/ssl/enable-internal-tls.yaml \
-e ${CUSTOM_TEMPLATES}/tls-parameters.yaml \
...
```

4. 通过查询 keystone 获取端点列表来确认每个端点正在使用 HTTPS：

```
openstack endpoint list
```

5.6. TRIPLEO-IPA 的参数

使用云的完全限定域名(FQDN)来定义 **tripleo-ipa** 所需的云名称和云域参数。例如，对于 FQDN 为 **overcloud.example.com**，使用以下值：

- CloudDomain: example.com
- cloudName: overcloud.example.com
- CloudNameCtlplane: overcloud.ctlplane.example.com
- CloudNameInternal: overcloud.internalapi.example.com
- CloudNameStorage: overcloud.storage.example.com
- CloudNameStorageManagement: overcloud.storagemgmt.example.com

根据您的环境要求设置以下附加参数：

CertmongerKerberosRealm

将 **CertmongerKerberosRealm** 参数设置为 IPA realm 的值。如果 IPA 域与 IPA 域不匹配，则需要此项。

DnsSearchDomains

DnsSearchDomains 参数是一个用逗号分开的列表。如果 IdM 服务器的域与云域不同，请在 **DnsSearchDomains** 参数中包含 IdM 服务器的域。

EnableEtcdInternalTLS

如果在分布式计算节点(DCN)架构上部署 TLSe，您必须添加 **EnableEtcdInternalTLS** 参数，值设为 **True**。

IDMInstallClientPackages

如果您预置备了计算节点，请将 **IDMInstallClientPackages** 参数设置为 **True** 的值。否则，将值设为 **False**。

IDMModifyDNS

将 **IDMModifyDNS** 参数设置为 **false**，以禁用 Red Hat Identity Server 上 overcloud 节点的自动 IP 注册。

IdmDomain

将 **IdmDomain** 参数设置为红帽身份服务器的 FQDN 的域部分。您指定的值也用作 IdM 域的值。如果 IdM 域和 IdM 域不同，请使用 **CertmongerKerberosRealm** 参数明确设置域。

IdmServer

将 **IdmServer** 参数设置为红帽身份服务器的 FQDN。如果您使用复制的 IdM 环境，请使用以逗号分隔的列表设置多个值。有关 IdM 副本的更多信息，请参阅安装 IdM 副本。

5.7. 在 TLS 下加密 MEMCACHED 流量(TLS-E)

现在，您可以使用 TLS-e 加密 memcached 流量。此功能可用于 novajoin 和 tripleo-ipa：

1. 创建名为 **memcached.yaml** 的环境文件，其内容如下，以添加对 memcached 的 TLS 支持：

```
parameter_defaults:
  MemcachedTLS: true
  MemcachedPort: 11212
```

2. 在 overcloud 部署过程中包含 **memcached.yaml** 环境文件：

```
openstack overcloud deploy --templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/enable-internal-tls.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-everywhere-endpoints-dns.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/haproxy-public-tls-certmonger.yaml \
-e /home/stack/memcached.yaml
...
```

其它资源

- 有关使用 tripleo-ipa 部署 TLSe 的更多信息，请参阅使用 Ansible 实施 TLS-e。

5.8. 增加私钥的大小

您可以通过增加为加密服务流量创建证书的私钥大小来提高安全性。默认 RHOSP 私钥大小为 2048 位，与美国国家标准与技术研究所(NIST)的要求匹配。

- 使用 **CertificateKeySize** 参数来全局更改私钥大小。
- 使用特定于服务的参数，如 **RedisCertificateKeySize**、修改特定私钥或覆盖全局 **CertificateKeySize** 参数。

在环境 heat 模板中使用这些参数，并在 overcloud 部署命令中包含模板。如果您已经部署了 overcloud，则必须使用您原来所用的相同模板重新运行相同的 **openstack overcloud deploy** 命令，并包含新参数才能使更改生效。

在以下示例中，私钥的全局值为 **4096**。**redis** 的私钥是 **2048**，因为 **RedisCertificateKeySize** 覆盖全局参数：

Example

```
parameter_defaults:
    CertificateKeySize: '4096'
    RedisCertificateKeySize: '2048'
```

5.9. 将 RED HAT OPENSTACK PLATFORM 的 IDM 服务器替换为其副本

当您将现有的 IPA 服务器替换为其副本时，您必须更新必要的参数。如果不这样做，会导致在更新集群配置时 overcloud 部署失败。

流程

1. 在每个 overcloud 节点上，编辑 **/etc/ipa/default.conf** 配置文件，以确保 **server** 和 **xmlrpc_uri** 参数包含 IdM 服务器的完全限定域名(FQDN)：

```
#File modified by ipa-client-install

[global]
basedn = dc=redhat,dc=local
realm = REDHAT.LOCAL
domain = redhat.local
server = freeipa-0.redhat.local
host = undercloud-0.redhat.local
xmlrpc_uri = https://freeipa-0.redhat.local/ipa/xml
enable_ra = True
```

2. 在 undercloud 上，编辑 **/home/stack/templates/tls-parameters.yaml** 环境文件，并确保 **IPA_SERVER_HOSTNAME** 参数与 **/etc/ipa/default.conf** 中的 **xmlrpc_uri** 和 server 参数中显示的 FQDN 匹配。确保所有参数都与您的环境匹配：

```
export IPA_DOMAIN=bigcorp.com
export IPA_REALM=BIGCORP.COM
export IPA_ADMIN_USER=$IPA_USER
export IPA_ADMIN_PASSWORD=$IPA_PASSWORD
export IPA_SERVER_HOSTNAME=ipa.bigcorp.com
export UNDERCLOUD_FQDN=undercloud.example.com
export USER=stack
export CLOUD_DOMAIN=example.com
```

第 6 章 配置自定义 SSL/TLS 证书

您可以手动配置 undercloud，以使用 SSL/TLS 通过公共端点进行通信。当您使用 SSL/TLS 手动配置 undercloud 端点时，您要创建安全端点作为概念验证。红帽建议使用证书颁发机构解决方案。

当使用证书颁发机构(CA)解决方案时，您有生产就绪的解决方案，如证书续订、证书撤销列表(CRL)和行业接受加密。有关使用 Red Hat Identity Manager (IdM)作为 CA 的详情，请参考使用 Ansible 实施 TLS-e。

如果要将 SSL 证书与您自己的证书颁发机构搭配使用，您必须完成以下配置步骤。

6.1. 初始化签名主机

签名主机是使用证书颁发机构生成并签名新证书的主机。如果您从未在所选签名主机上创建 SSL 证书，您可能需要初始化该主机，让它能够为新证书签名。

流程

1. **/etc/pki/CA/index.txt** 文件包含所有签名证书的记录。请确定文件系统路径和 **index.txt** 文件已存在：

```
$ sudo mkdir -p /etc/pki/CA
$ sudo touch /etc/pki/CA/index.txt
```

2. **/etc/pki/CA/serial** 文件标识下一个序列号，以用于下一个要签名的证书。检查是否存在此文件。如果此文件不存在，则使用新启动值创建新文件：

```
$ echo '1000' | sudo tee /etc/pki/CA/serial
```

6.2. 创建证书颁发机构

一般情况下，您需要使用一个外部的证书认证机构来签发您的 SSL/TLS 证书。在某些情况下，您可能想使用自己的证书颁发机构。例如，您可能想拥有仅限内部使用的证书颁发机构。

流程

1. 生成密钥和证书对以充当证书颁发机构：

```
$ openssl genrsa -out ca.key.pem 4096
$ openssl req  -key ca.key.pem -new -x509 -days 7300 -extensions v3_ca -out ca.crt.pem
```

2. **openssl req** 命令会要求输入认证机构的详细信息。根据提示输入相关信息。这些命令创建一个称为 **ca.crt.pem** 的证书颁发机构文件。

3. 在 **enable-tls.yaml** 文件中将证书位置设置为 **PublicTLSCAFile** 参数的值。当您将证书位置设置为 **PublicTLSCAFile** 参数的值时，您可以确保将 CA 证书路径添加到 **clouds.yaml** 身份验证文件中。

```
parameter_defaults:
  PublicTLSCAFile: /etc/pki/ca-trust/source/anchors/cacert.pem
```

6.3. 将此证书颁发机构添加到客户端

对于旨在使用 SSL/TLS 通信的所有外部客户端，将证书颁发机构文件复制到需要访问 Red Hat OpenStack Platform 环境的每个客户端。

流程

1. 将证书颁发机构复制到客户端系统：

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

2. 将证书颁发机构文件复制到每个客户端后，在每个客户端上运行以下命令，将证书添加到证书颁发机构信任捆绑包中：

```
$ sudo update-ca-trust extract
```

6.4. 创建 SSL/TLS 密钥

在 OpenStack 环境中启用 SSL/TLS 需要一个 SSL/TLS 密钥来生成证书。

流程

1. 运行以下命令以生成 SSL/TLS 密钥 (**server.key.pem**)：

```
$ openssl genrsa -out server.key.pem 2048
```

6.5. 创建 SSL/TLS 证书签名请求

完成以下步骤以创建证书签名请求。

流程

1. 复制默认 OpenSSL 配置文件：

```
$ cp /etc/pki/tls/openssl.cnf .
```

2. 编辑新的 **openssl.cnf** 文件并配置要用于 director 的 SSL 参数。一个要修改的参数类型的示例包括：

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req

[req_distinguished_name]
countryName = Country Name (2 letter code)
countryName_default = AU
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Queensland
localityName = Locality Name (eg, city)
localityName_default = Brisbane
organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = Red Hat
commonName = Common Name
commonName_default = 192.168.0.1
```

```
commonName_max = 64

[ v3_req ]
# Extensions to add to a certificate request
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names

[alt_names]
IP.1 = 192.168.0.1
DNS.1 = instack.localdomain
DNS.2 = vip.localdomain
DNS.3 = 192.168.0.1
```

将 **commonName_default** 设置为以下条目之一：

- 如果使用 IP 地址通过 SSL/TLS 访问 director，则使用 **undercloud.conf** 文件中的 **undercloud_public_host** 参数。
- 如果使用完全限定域名通过 SSL/TLS 访问 director，则使用此域名。

编辑 **alt_names** 部分，使其包含以下条目：

- **IP** - 客户端用于通过 SSL 访问 director 的 IP 地址列表。
- **DNS** - 客户端用于通过 SSL 访问 director 的域名列表。其中也包含公共 API IP 地址作为在 **alt_names** 部分末尾的 DNS 条目。

注意

有关 **openssl.cnf** 的更多信息，请运行 **man openssl.cnf** 命令。

3. 运行以下命令以生成证书签名请求 (**server.csr.pem**)：

```
$ openssl req -config openssl.cnf -key server.key.pem -new -out server.csr.pem
```

确保使用 **-key** 选项包括 OpenStack SSL/TLS 密钥。

此命令生成 **server.csr.pem** 文件，这是证书签名请求。使用此文件创建 OpenStack SSL/TLS 证书。

6.6. 创建 SSL/TLS 证书

要为 OpenStack 环境生成 SSL/TLS 证书，必须存在以下文件：

openssl.cnf
指定 v3 扩展的自定义配置文件。

server.csr.pem
生成证书并使用 CA 对证书进行签名的证书签名请求。

ca.crt.pem
对证书进行签名的证书颁发机构。

ca.key.pem
证书颁发机构私钥。

流程

1. 如果尚未存在，请创建 **newcerts** 目录：

```
sudo mkdir -p /etc/pki/CA/newcerts
```

2. 运行以下命令，为 undercloud 或 overcloud 创建证书：

```
$ sudo openssl ca -config openssl.cnf -extensions v3_req -days 3650 -in server.csr.pem -out
server.crt.pem -cert ca.crt.pem -keyfile ca.key.pem
```

这个命令使用以下选项：

-config
使用一个自定义配置文件，它是带有 v3 扩展的 **openssl.cnf** 文件。

-extensions v3_req
已启用的 v3 扩展。

-days
定义证书到期前的天数。

-in
证书签名请求。

-out
生成的签名证书。

-cert
证书颁发机构文件。

-keyfile
证书颁发机构私钥。

此命令创建名为 **server.crt.pem** 的新证书。将此证书与 OpenStack SSL/TLS 密钥一起使用

6.7. 将证书添加到 UNDERCLOUD

完成以下步骤，将 OpenStack SSL/TLS 证书添加到 undercloud 信任捆绑包中。

流程

1. 运行以下命令以组合证书和密钥：

```
$ cat server.crt.pem server.key.pem > undercloud.pem
```

此命令创建 **undercloud.pem** 文件。

2. 将 **undercloud.pem** 文件复制到 **/etc/pki** 目录内的位置，并设置必要的 SELinux 上下文，以便 HAProxy 能够读取：

```
$ sudo mkdir /etc/pki/undercloud-certs
$ sudo cp ~/undercloud.pem /etc/pki/undercloud-certs/.
$ sudo semanage fcontext -a -t etc_t "/etc/pki/undercloud-certs(/.*)?"
$ sudo restorecon -R /etc/pki/undercloud-certs
```

3. 将 **undercloud.pem** 文件位置添加到 **undercloud.conf** 文件的 **undercloud_service_certificate** 选项中：

```
undercloud_service_certificate = /etc/pki/undercloud-certs/undercloud.pem
```

不要设置或启用 **generate_service_certificate** 和 **certificate_generation_ca** 参数。director 使用这些参数自动生成证书，而不使用您手动创建的 **undercloud.pem** 证书。

4. 将签名证书的证书颁发机构添加到 undercloud 的可信证书颁发机构列表中，以便 undercloud 内的不同服务能够访问证书颁发机构：

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

要验证证书颁发机构是否已添加到 undercloud，请使用 **openssl** 检查信任捆绑包：

```
$ openssl crl2pkcs7 -nocrl -certfile /etc/pki/tls/certs/ca-bundle.crt | openssl pkcs7 -print_certs
-text | grep <CN of the CA issuer> -A 10 -B 10
```

将 **<CN of the CA issuer>** 替换为 CA 颁发者的通用名称。此命令输出主要证书详细信息，包括有效期日期。

第 7 章 在 OVERCLOUD 公共端点中启用 SSL/TLS

默认情况下，overcloud 将未加密的端点用于 overcloud 服务。要在 overcloud 中启用 SSL/TLS，红帽建议您使用证书颁发机构(CA)解决方案。

使用 CA 解决方案时，您有生产就绪的解决方案，如证书续订、证书撤销列表(CRL)和行业接受加密。有关使用 Red Hat Identity Manager (IdM)作为 CA 的详情，请参考使用 Ansible 实施 TLS-e。

您可以使用以下手动过程来为公共 API 端点启用 SSL/TLS，内部和 Admin API 仍保留未加密的。如果不使用 CA，还必须手动更新 SSL/TLS 证书。如需更多信息，请参阅 手动更新 SSL/TLS 证书。

先决条件

- 定义公共 API 的端点的网络隔离。
- **openssl-perl** 软件包已安装。
- 您有一个 SSL/TLS 证书。如需更多信息，请参阅 配置自定义 SSL/TLS 证书。

7.1. 启用 SSL/TLS

要在 overcloud 中启用 SSL/TLS，您必须创建一个环境文件，其中包含 SSL/TLS 证书和私钥的参数。

流程

1. 从 heat 模板集合中复制 **enable-tls.yaml** 环境文件：

```
$ cp -r /usr/share/openstack-tripleo-heat-templates/environments/ssl/enable-tls.yaml
~/templates/.
```

2. 编辑此文件并为这些参数进行以下更改：

SSLCertificate

将证书文件的内容(**server.crt.pem**)复制到 **SSLCertificate** 参数中：

```
parameter_defaults:
  SSLCertificate: |
    -----BEGIN CERTIFICATE-----
    MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGS
    ...
    sFW3S2roS4X0Af/kSSD8mlBBTFTCMBAj6rtLBKLaQ
    -----END CERTIFICATE-----
```

重要

证书内容为所有新行需要相同的缩进级别。

SSLIntermediateCertificate

如果您有中间证书，请将中间证书的内容复制到 **SSLIntermediateCertificate** 参数中：

```
parameter_defaults:
  SSLIntermediateCertificate: |
```

```
-----BEGIN CERTIFICATE-----
sFW3S2roS4X0Af/kSSD8mlBBTFTCMBAj6rtLBKLaQbIxEpIzrgvpBCwUAMFgxCzAJB
...
MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGSIb3DQE
-----END CERTIFICATE-----
```

重要

证书内容为所有新行需要相同的缩进级别。

SSLKey

将私钥的内容(**server.key.pem**)复制到 **SSLKey** 参数中：

```
parameter_defaults:
  ...
  SSLKey: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIEowIBAAKCAQEAqVw8lnQ9RbeI1EdLN5PJP0IVO
    ...
    ctlKn3rAAdyumi4JDjESAXHIKFjJNOLrBmpQyES4X
    -----END RSA PRIVATE KEY-----
```

重要

私钥内容为所有新行需要相同的缩进级别。

7.2. 注入 ROOT 证书

如果证书签名者不在 overcloud 镜像上的默认信任存储中，您必须将证书颁发机构注入 overcloud 镜像中。

流程

1. 从 heat 模板集合中复制 **inject-trust-anchor-hiera.yaml** 环境文件：

```
$ cp -r /usr/share/openstack-tripleo-heat-templates/environments/ssl/inject-trust-anchor-hiera.yaml ~/templates/.
```

编辑此文件并为这些参数进行以下更改：

CAMap

列出要注入到 overcloud 的每个证书颁发机构内容(CA)。overcloud 要求用于为 undercloud 和 overcloud 为证书签名 CA 文件。将 root 证书颁发机构文件(**ca.crt.pem**)的内容复制到条目中。例如，您的 **CAMap** 参数可能类似如下：

```
parameter_defaults:
  CAMap:
    ...
    undercloud-ca:
      content: |
```

```
    -----BEGIN CERTIFICATE-----
    MIIDITCCAn2gAwIBAgIJAOnPtx2hHEhrMA0GCS
    BAYTAlVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBw
    UmVkIEhhdDELMAkGA1UECwwCUUUxFDASBgNVBA
    -----END CERTIFICATE-----
overcloud-ca:
  content: |
    -----BEGIN CERTIFICATE-----
    MIIDBzCCAe+gAwIBAgIJAIc75A7FD++DMA0GCS
    BAMMD3d3dy5leGFtcGxlLmNvbTAeFw0xOTAxMz
    Um54yGCARyp3LpkxvyfMXX1DokpS1uKi7s6CkF
    -----END CERTIFICATE-----
```

重要

证书颁发机构内容为所有新行需要相同的缩进级别。

您还可以使用 **CAMap** 参数注入其他 CA。

7.3. 配置 DNS 端点

如果您使用 DNS 主机名通过 SSL/TLS 访问 overcloud，请将 **/usr/share/openstack-tripleo-heat-templates/environments/predictable-placement/custom-domain.yaml** 文件复制到 **/home/stack/templates** 目录中。

注意

如果初始部署中没有包含此环境文件，则无法使用 TLS-everywhere 架构重新部署。

为所有字段配置主机和域名，根据需要为自定义网络添加参数：

CloudDomain

主机的 DNS 域。

CloudName

overcloud 端点的 DNS 主机名。

CloudNameCtlplane

provisioning 网络端点的 DNS 名称。

CloudNameInternal

内部 API 端点的 DNS 名称。

CloudNameStorage

存储端点的 DNS 名称。

CloudNameStorageManagement

存储管理端点的 DNS 名称。

流程

- 使用以下参数之一添加要使用的 DNS 服务器：
  - **DEFAULT/undercloud_nameservers**

- **%SUBNET_SECTION%/dns_nameservers**

提示

您可以使用 **CloudName{network.name}** 定义，在使用虚拟 IP 的可组合网络上为 API 端点设置 DNS 名称。

如需更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 中的 添加可组合网络。

7.4. 在 OVERCLOUD 创建过程中添加环境文件

将 **-e** 选项与部署命令 **openstack overcloud deploy** 搭配使用，以在部署过程中包含环境文件。按照以下顺序添加本节中的环境文件：

- 启用 SSL/TLS 的环境文件(**enable-tls.yaml**)
- 用于设置 DNS 主机名(**custom-domain.yaml**)的环境文件
- 注入 root 证书颁发机构的环境文件(**inject-trust-anchor-hiera.yaml**)
- 设置公共端点映射的环境文件：
  - 如果您使用 DNS 名称来访问公共端点，请使用 **/usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-dns.yaml**
  - 如果您使用 IP 地址访问公共端点，请使用 **/usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-ip.yaml**

流程

- 使用以下部署命令片断作为如何包含 SSL/TLS 环境文件的示例：

```
$ openstack overcloud deploy --templates \
[...]
-e /home/stack/templates/enable-tls.yaml \
-e ~/templates/custom-domain.yaml \
-e ~/templates/inject-trust-anchor-hiera.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-dns.yaml
```

7.5. 手动更新 SSL/TLS 证书

如果您使用没有从 TLS 随处(TLS-e)过程自动生成的您自己的 SSL/TLS 证书，请完成以下步骤。

流程

1. 使用以下内容编辑 heat 模板：
   - 编辑 **enable-tls.yaml** 文件并更新 **SSLCertificate**、**SSLKey** 和 **SSLIntermediateCertificate** 参数。
   - 如果您的证书颁发机构已更改，请编辑 **inject-trust-anchor-hiera.yaml** 文件并更新 **CAMap** 参数。
2. 重新运行部署命令：

```
$ openstack overcloud deploy --templates \
[...]
-e /home/stack/templates/enable-tls.yaml \
-e ~/templates/custom-domain.yaml \
-e ~/templates/inject-trust-anchor-hiera.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-dns.yaml
```

3. 在 director 上，为每个 Controller 运行以下命令：

```
ssh tripleo-admin@<controller> sudo podman \
restart $(podman ps --format="{{.Names}}" | grep -w -E 'haproxy(-bundle-.*-[0-9]+)?')
```

第 8 章 在 OVERCLOUD 中使用 FERNET 密钥进行加密

Fernet 是默认的令牌提供程序，它取代 **uuid**。您可以查看 Fernet 部署并测试令牌是否正常工作。Fernet 使用三种类型的密钥，它们存储在 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/fernet-keys** 中。最高数字目录包含主密钥，它生成新的令牌并解密现有的令牌。

8.1. 检查 FERNET 部署

要测试 Fernet 令牌是否正常工作，请检索 Controller 节点的 IP 地址，SSH 到 Controller 节点，然后检查令牌驱动程序和提供程序的设置。

流程

1. 检索 Controller 节点的 IP 地址：

```
[stack@director ~]$ source ~/stackrc
[stack@director ~]$ openstack server list
-------------------------------------------------------------------------------------------+
| ID                                    | Name                     | Status | Networks          |
-------------------------------------------------------------------------------------------+
| 756fbd73-e47b-46e6-959c-e24d7fb71328 | overcloud-controller-0  | ACTIVE |
ctlplane=192.0.2.16 |
| 62b869df-1203-4d58-8e45-fac6cd4cfbee | overcloud-novacompute-0 | ACTIVE |
ctlplane=192.0.2.8  |
-------------------------------------------------------------------------------------------+
```

2. SSH 到 Controller 节点：

```
[tripleo-admin@overcloud-controller-0 ~]$ ssh tripleo-admin@192.0.2.16
```

3. 检索令牌驱动程序和提供程序设置的值：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo crudini --get /var/lib/config-data/puppet-
generated/keystone/etc/keystone/keystone.conf token driver
sql
[tripleo-admin@overcloud-controller-0 ~]$ sudo crudini --get /var/lib/config-data/puppet-
generated/keystone/etc/keystone/keystone.conf token provider
fernet
```

4. 测试 Fernet 供应商：

```
[tripleo-admin@overcloud-controller-0 ~]$ exit
[stack@director ~]$ source ~/overcloudrc
[stack@director ~]$ openstack token issue
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------+
| Field | Value |
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------+
| expires | 2016-09-20 05:26:17+00:00 |
| id | gAAAAABX4LppE8vaiFZ992eah2i3edpO1aDFxlKZq6a_RJzxUx56QVKORrmW0-oZK3-
Xuu2wcnpYq_eek2SGLz250eLpZOzxKBR0GsoMfxJU8mEFF8NzfLNcbuS-iz7SV-
N1re3XEywSDG90JcgwjQfXW-8jtCm-n3LL5IaZexAYIw059T_-cd8 |
```

```
| project_id | 26156621d0d54fc39bf3adb98e63b63d |
| user_id | 397daf32cadd490a8f3ac23a626ac06c |
----------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
```

结果包括长的 Fernet 令牌。

第 9 章 RED HAT OPENSTACK PLATFORM 上的联邦信息处理标准

联邦信息处理标准(FIPS)是由国家标准与技术研究所开发的一组安全要求(NIST)。在 Red Hat Enterprise Linux 9 中，支持的标准是 FIPS 出版物 140-3，*加密模块的安全要求*。有关支持的标准的详情，请查看联邦信息处理标准 140-3。

这些安全要求定义可接受的加密算法，以及使用这些加密算法，包括安全模块。

- FIPS 140-3 验证是通过 FIPS 批准的加密算法来实现的，使用规定的方式通过验证的模块获得。
- FIPS 140-3 兼容性只能通过 FIPS 批准的加密算法来实现。

Red Hat OpenStack Platform 17 是 **FIPS 140-3 兼容**。您可以使用红帽提供的镜像来部署 overcloud，利用 FIPS 兼容性。

注意

OpenStack 17.1 基于 Red Hat Enterprise Linux (RHEL) 9.2。RHEL 9.2 尚未提交用于 FIPS 验证。红帽计划获取 RHEL 9.0 和 RHEL 9.2 以及以后的 RHEL 9.x 偶数的次版本模块的 FIPS 验证，但没有具体的时间表。更新将包括在 Compliance Activities and Government Standards 中。

9.1. 启用 FIPS

启用 FIPS 后，您必须在安装 undercloud 和 overcloud 的过程中完成一系列步骤。

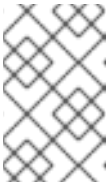
先决条件

- 已安装 Red Hat Enterprise Linux，并准备好开始安装 Red Hat OpenStack Platform director。
- 如果您使用 Red Hat Ceph Storage 作为存储后端，则部署 Red Hat Ceph Storage 6 或更高版本。

流程

1. 在 undercloud 上启用 FIPS：
   a. 在您要在其上安装 undercloud 的系统中启用 FIPS：

```
fips-mode-setup --enable
```



注意

此步骤会将 **fips=1** 内核参数添加到 GRUB 配置文件中。因此，只有 Red Hat Enterprise Linux 使用的加密算法模块处于 FIPS 模式，且只使用标准批准的加密算法。

   b. 重启系统：
   c. 验证是否启用了 FIPS：

```
fips-mode-setup --check
```

d. 安装和配置 Red Hat OpenStack Platform director。有关更多信息，请参阅在 undercloud 上安装 director。

2. 为 overcloud 准备启用了 FIPS 的镜像。

a. 为 overcloud 安装镜像：

```
sudo dnf -y install rhosp-director-images-uefi-fips-x86_64
```

b. 在 **stack** 用户的主目录中创建 **images** 目录：

```
$ mkdir /home/stack/images
$ cd /home/stack/images
```

c. 将镜像提取到您的主目录中：

```
for i in /usr/share/rhosp-director-images/*fips*.tar; do tar -xvf $i; done
```

d. 在上传镜像前，您必须创建符号链接：

```
ln -s ironic-python-agent-fips.initramfs       ironic-python-agent.initramfs
ln -s ironic-python-agent-fips.kernel          ironic-python-agent.kernel
ln -s overcloud-hardened-uefi-full-fips.qcow2  overcloud-hardened-uefi-full.qcow2
```

e. 将启用了 FIPS 的 overcloud 镜像上传到镜像服务：

```
openstack overcloud image upload --update-existing --whole-disk
```

注意

即使当前没有 OpenStack 镜像服务中的镜像，您必须使用 **--update-existing** 标志。

3. 在 overcloud 上启用 FIPS。
为特定于您的环境的 overcloud 部署配置模板。在部署命令中包含所有配置模板，包括 fips.yaml：

```
openstack overcloud deploy
...
-e /usr/share/openstack-tripleo-heat-templates/environments/fips.yaml
```

第 10 章 提高用户访问安全性

您可以在 Red Hat OpenStack Platform 17 中启用安全基于角色的访问控制(SRBAC)。SRBAC 模型有三个用户角色，它基于项目范围内现有的三个角色。

10.1. SRBAC 用户角色

用户角色是角色及其所属范围的组合。部署 Red Hat OpenStack Platform 17 时，您可以从项目范围内分配任何用户角色。

10.1.1. Red Hat OpenStack Platform SRBAC 角色

目前，项目范围内提供了三个不同的角色。

admin

admin 角色包括资源或 API 的所有创建、读取、更新或删除操作。

成员

允许 **member** 角色创建、读取、更新和删除属于成员范围的资源。

读取器

reader 角色用于只读操作，无论它要应用到的范围。此角色可以查看应用它的整个范围的资源。

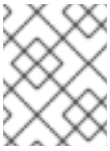
10.1.2. Red Hat OpenStack Platform SRBAC 范围

范围是执行操作的上下文。Red Hat OpenStack Platform 17 中仅提供 **项目范围**。**项目** 范围是 OpenStack 中隔离自助服务资源的 API 部分。

10.1.3. Red Hat OpenStack Platform SRBAC 用户角色

项目管理员

由于项目管理员用户角色是唯一可用的管理人员，因此 Red Hat OpenStack Platform 17 包括了修改的策略，为项目管理员授予最高级别授权。此 persona 包括在项目间创建、读取、更新和删除操作，其中包括添加和删除用户和其他项目。



注意

该人员预计会随着未来的开发范围而有所变化。此角色表示授予项目成员和项目读取器的所有权限。

项目成员

项目成员用户角色适用于被授予在项目范围内消耗资源权限的用户。此用户角色可以在为其分配的项目中创建、列出、更新和删除资源。此用户角色表示授予项目读取器的所有权限。

项目读取器

项目读取器用于授予查看项目中非敏感资源的权限的用户。在项目中，为需要检查或查看资源或审核员的用户分配 reader 角色，他们只需要查看单个项目中的特定于项目的资源，以满足审计目的。project-reader 人员不会解决所有审计用例。

基于 **system** 或 **domain** 范围的额外人员正在开发中，还不可用。

注意

镜像服务(glance)不支持 metadef API 的 SRBAC 权限。RHOSP 17.1 中用于镜像服务 metadef API 的默认策略仅用于 admin。

10.2. 激活安全基于角色的访问控制

当您激活基于角色的安全身份验证时，您将激活一组新的策略文件，该文件定义在 Red Hat OpenStack Platform 环境中分配给用户的权限范围。

先决条件

- 已安装 Red Hat OpenStack Platform director 环境。

流程

- 在部署 Red Hat OpenStack Platform 时，在部署脚本中包含 **enable-secure-rbac.yaml** 环境文件：

```
openstack overcloud deploy --templates
…
-e /usr/share/openstack-tripleo-heat-templates/environments/enable-secure-rbac.yaml
```

10.3. 在 SRBAC 环境中分配角色

通过 Red Hat OpenStack Platform 上的 SRBAC，您可以将用户分配给 **project-admin**、**project-member** 或 **project-reader** 的角色。

先决条件

- 您已部署了具有基于安全角色的身份验证(SRBAC)的 Red Hat OpenStack Platform。

流程

- 使用以下语法，使用 **openstack role add** 命令：
  - 分配 **admin** 角色：

```
$ openstack role add --user <user> --user-domain <domain> --project <project> --project-domain <project-domain> admin
```

  - 分配 **member** 角色：

```
$ openstack role add --user <user> --user-domain <domain> --project <project>  --project-domain <project-domain> member
```

  - 分配 **reader** 角色：

```
$ openstack role add --user <user> --user-domain <domain> --project <project> --project-domain <project-domain> reader
```

- 将 **<user** > 替换为要应用角色的现有用户。

- 将 **<** domain> 替换为角色应用到的域。
- 将 **&lt**;project> 替换为被授予该角色的项目。
- 将 **<project-domain** > 替换为项目所在域。

第 11 章 策略（POLICY）

每个 OpenStack 服务都包含由访问策略管理的资源。例如，资源可能包括以下功能：

- 创建和启动实例的权限
- 将卷附加到实例的功能

如果您是 Red Hat OpenStack Platform (RHOSP)管理员，您可以创建自定义策略来引入具有不同访问级别的新角色，或更改现有角色的默认行为。

重要

红帽不支持自定义角色或策略。语法错误或应用错误授权可能会对安全或可用性造成负面影响。如果在生产环境中需要自定义角色或策略，请联系红帽支持例外。

11.1. 查看现有策略

通常位于 **/etc/$service** 目录中服务的策略文件。例如，Compute (nova)的 **policy.json** 文件的完整路径是 **/etc/nova/policy.json**。

有两个重要的架构更改会影响如何查找现有策略：

- Red Hat OpenStack Platform 现已容器化。
  - 如果您从服务容器内查看，策略文件位于传统路径中：
    /etc/$service/policy.json
  - 如果您从服务容器外部查看，策略文件位于以下路径中：
    /var/lib/config-data/puppet-generated/$service/etc/$service/policy.json
- 每个服务都有在代码中提供的默认策略，其中文件只在手动创建文件时可用，或者使用 **oslopolicy** 工具生成这些文件。要生成策略文件，请使用容器中的 **oslopolicy-policy-generator**，如下例所示：

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

默认情况下，生成的策略由 oslo.policy CLI 工具推送到 stdout。

11.2. 了解服务策略

服务策略文件语句是别名定义或规则。别名定义存在于文件的顶部。以下列表包含 Compute (nova)生成的 **policy.json** 文件中的别名定义说明：

- "context_is_admin": "role:admin"
  当 **rule:context_is_admin** 在目标后出现时，策略会检查用户是否在允许该操作前使用管理上下文运行。
- "admin_or_owner": "is_admin:True or project_id:%(project_id)s"
  当 **admin_or_owner** 在目标后出现时，策略会检查用户是否为 admin，或者其项目 ID 与目标对象自己的项目 ID 匹配，然后再允许该操作。
- "admin_api": "is_admin:True
  当 **admin_api** 在目标后显示时，策略会检查用户是否为 admin，然后它允许该操作。

11.3. 策略语法

policy.json 文件支持某些 Operator，以便您可以控制这些设置的目标范围。例如，以下 keystone 设置包含只有 admin 用户可以创建用户的规则：

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

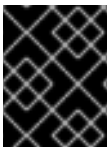
: 字符左侧的 部分描述了特权，而右侧的部分则定义谁可以使用特权。您还可以使用右侧的 Operator 来进一步控制范围：

- **!** - 没有用户（包括 admin）可以执行此操作。
- **@** 和 **""** - 任何用户可以执行此操作。
- **not**, **and**, **or** - 可以使用标准的操作符。

例如，以下设置意味着没有用户创建新用户的权限：

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

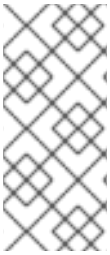
11.4. 使用策略文件进行访问控制



重要

红帽不支持自定义角色或策略。语法错误或应用错误授权可能会对安全或可用性造成负面影响。如果在生产环境中需要自定义角色或策略，请联系红帽支持例外。

要覆盖默认规则，请编辑相应 OpenStack 服务的 **policy.json** 文件。例如，Compute 服务在 nova 目录中有一个 **policy.json**，这是从容器内查看容器化服务的文件正确位置。



注意

- 您必须先全面测试暂存环境中对策略文件的更改，然后才能在生产环境中实施它们。
- 您必须检查对访问控制策略的任何更改不会意外地弱任何资源的安全性。另外，对 **policy.json** 文件的任何更改都会立即有效，不需要重启服务。

示例：创建高级用户角色

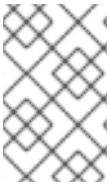
要自定义 keystone 角色的权限，请更新服务的 **policy.json** 文件。这意味着您可以更精细地定义分配给您类用户的权限。本例使用以下权限为部署创建一个 *power 用户角色* ：

- 启动一个实例。
- 停止实例。
- 管理附加到实例的卷。

此角色旨在向某些用户授予额外权限，而无需授予 **admin** 访问权限。要使用这些权限，您必须为自定义角色授予以下权限：

- 启动一个实例：" **os_compute_api:servers:start": "role:PowerUsers"**

- 停止实例：" **os_compute_api:servers:stop": "role:PowerUsers"**
- 配置实例以使用特定卷：" **os_compute_api:servers:create:attach_volume": "role:PowerUsers"**
- 列出附加到实例的卷：" **os_compute_api:os-volumes-attachments:index": "role:PowerUsers"**
- Attach a volume: **"os_compute_api:os-volumes-attachments:create": "role:PowerUsers"**
- 查看附加卷的详情：" **os_compute_api:os-volumes-attachments:show": "role:PowerUsers"**
- 更改附加到实例的卷：" **os_compute_api:os-volumes-attachments:update": "role:PowerUsers"**
- 删除附加到实例的卷：" **os_compute_api:os-volumes-attachments:delete": "role:PowerUsers"**



注意

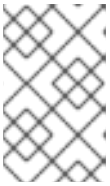
修改 **policy.json** 文件时，您可以覆盖默认策略。因此，**PowerUsers** 的成员是可以执行这些操作的唯一用户。要允许 **admin** 用户保留这些权限，您可以为 *admin_or_power_user 创建规则*。您还可以使用一些基本条件逻辑来定义 **role:PowerUsers 或 role:Admin**。

1. 创建自定义 keystone 角色：

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

2. 将现有用户添加到角色中，并将角色分配给项目：

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



注意

角色分配仅与一个项目配对。这意味着，当您为用户分配角色时，您也同时定义目标项目。如果您希望用户收到同一角色但针对不同的项目，则必须再次为它们分配角色，但以不同的项目为目标。

3. 查看默认的 nova 策略设置：

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

4. 通过向 **/var/lib/config-data/puppet-generated/nova/etc/nova/policy.json** 中添加以下条目，为新的 **PowerUsers** 角色创建自定义权限：

注意

在部署前测试您的策略更改，以验证它们是否按预期工作。

```
{
"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"
}
```

在保存此文件时，您可以实施更改并重新启动 nova 容器。添加到 **PowerUsers** keystone 角色的用户会收到这些权限。

11.5. 示例：根据属性限制访问

重要

红帽不支持自定义角色或策略。语法错误或应用错误授权可能会对安全或可用性造成负面影响。如果在生产环境中需要自定义角色或策略，请联系红帽支持例外。

您可以创建策略，以根据发出该 API 调用的用户属性限制对 API 调用的访问。例如，如果从管理上下文运行，或者令牌的用户 ID 与与目标关联的用户 ID 匹配，则允许以下默认规则删除密钥对。

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

注意：* 新实现的功能无法保证在每个发行版本的每个服务中。因此，使用目标服务的现有策略惯例编写规则非常重要。有关查看这些策略的详情，请参阅 检查现有策略。* 所有策略都应针对要部署它们的每个版本在非生产环境中测试，因为策略无法保证跨发行版本的兼容性。

根据上例，您可以制作 API 规则来根据用户是否拥有资源来扩展或限制对用户的访问。另外，属性也可以与其他限制相结合来组成规则，如下例所示：

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

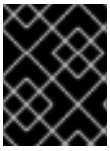
在上面的示例中，您可以创建一个只限于管理员和用户的唯一规则，然后使用该规则进一步限制操作：

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

其他资源

- 策略语法.

11.6. 使用 HEAT 修改策略



重要

红帽不支持自定义角色或策略。语法错误或应用错误授权可能会对安全或可用性造成负面影响。如果在生产环境中需要自定义角色或策略，请联系红帽支持例外。

您可以使用 heat 为 overcloud 中的某些服务配置访问策略。使用以下参数在相应服务上设置策略：

表 11.1. 策略参数

参数	描述
KeystonePolicies	为 OpenStack Identity (keystone)配置的策略哈希。
IronicApiPolicies	为 OpenStack Bare Metal (ironic) API 配置的策略哈希。
BarbicanPolicies	为 OpenStack Key Manager (barbican)配置的策略哈希。
NeutronApiPolicies	为 OpenStack Networking (neutron) API 配置的策略哈希。
SaharaApiPolicies	为 OpenStack 集群(sahara) API 配置的策略哈希。
NovaApiPolicies	为 OpenStack Compute (nova) API 配置的策略哈希。
CinderApiPolicies	为 OpenStack Block Storage (cinder) API 配置的策略哈希。
GlanceApiPolicies	为 OpenStack Image Storage (glance) API 配置的策略哈希。
HeatApiPolicies	为 OpenStack Orchestration (heat) API 配置的策略哈希。

要为服务配置策略，请为 policy 参数指定一个包含服务策略的哈希值，例如：

- OpenStack Identity (keystone)使用 **KeystonePolicies** 参数。在环境文件的 **parameter_defaults** 部分中设置此参数：

```
parameter_defaults:
  KeystonePolicies: { keystone-context_is_admin: { key: context_is_admin, value: 'role:admin' } }
```

- OpenStack Compute (nova)使用 **NovaApiPolicies** 参数。在环境文件的 **parameter_defaults** 部分中设置此参数：

```
parameter_defaults:
  NovaApiPolicies: { nova-context_is_admin: { key: 'compute:get_all', value: '@' } }
```

11.7. 审计您的用户和角色

您可以使用 Red Hat OpenStack Platform 中提供的工具，为每个用户和相关特权构建角色分配的报告。

先决条件

- 已安装 Red Hat OpenStack Platform 环境。
- 以 stack 身份登录 director。

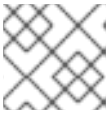
流程

1. 运行 **openstack role list** 命令以查看您环境中当前的角色：

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

swiftoperator
ResellerAdmin
admin
_member_
heat_stack_user
```

2. 运行 **openstack role assignment list** 命令，以列出属于特定角色的所有用户。例如，要查看具有 admin 角色的所有用户，请运行以下命令：

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



注意

您可以使用 **-f {csv,json,table,value,yaml}** 参数导出这些结果。

11.8. 审计 API 访问

您可以审核给定角色的 API 调用可以访问。为每个角色重复此过程将导致全面报告每个角色的可访问 API。

先决条件

- 作为目标角色中的用户提供的身份验证文件。
- JSON 格式的访问令牌。
- 您希望审核的每个服务的 API 的策略文件。

流程

1. 首先，在所需角色中提供用户的身份验证文件。
2. 捕获 Keystone 生成的令牌并将其保存到文件中。您可以通过运行任何 openstack-cli 命令并使用 --debug 选项进行此操作，该选项会将提供的令牌输出到 stdout。您可以复制此令牌并将其保存到访问文件中。使用以下命令作为单一步骤进行此操作：

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

3. 创建策略文件。这可以在托管所感兴趣的容器化服务的 overcloud 节点上完成。以下示例为 cinder 服务创建一个策略文件：

```
ssh tripleo-admin@CONTROLLER-1 sudo podman exec cinder_api \
oslopolicy-policy-generator \
--config-file /etc/cinder/cinder.conf \
--namespace cinder > cinderpolicy.json
```

4. 使用这些文件，您现在可以审核对 cinder API 的访问问题的角色：

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

第 12 章 网络时间协议

您需要确保 Red Hat OpenStack Platform 集群中的系统有准确且一致的系统时间戳。

Red Hat OpenStack Platform on Red Hat Enterprise Linux 9 支持 Chrony 进行时间管理。如需更多信息，请参阅使用 Chrony 套件配置 NTP。

12.1. 为什么一致性时间非常重要

整个机构的一致时间对于操作和安全需求至关重要：

识别安全事件

一致的计时可帮助您关联受影响系统上事件的时间戳，以便您可以了解事件序列。

身份验证和安全系统

安全系统可能对时间偏移非常敏感，例如：

- 基于 kerberos 的身份验证系统可能会拒绝验证受时钟偏移影响的客户端。
- 传输层安全性(TLS)证书取决于有效的时间源。如果客户端和服务器系统时间超过 *Valid From* 日期范围，则客户端到服务器 TLS 连接的客户端会失败。

Red Hat OpenStack Platform 服务

某些核心 OpenStack 服务特别依赖于准确计时，包括高可用性(HA)和 Ceph。

12.2. NTP 设计

网络时间协议(NTP)以分级设计进行组织。每个层称为 stratum。在层次结构的顶部是 stratum 0 设备，如原子时钟。在 NTP 层次结构中，stratum 0 设备为公开可用的 stratum 1 和 stratum 2 NTP 时间服务器提供参考。

请勿将您的数据中心客户端直接连接到公开可用的 NTP stratum 1 或 2 服务器。直接连接数量会给公共 NTP 资源带来不必要的。相反，请在数据中心中分配一个专用时间服务器，并将客户端连接到该专用服务器。

配置实例以从您的专用时间服务器接收时间，而不是它们所在的主机。

注意

在 Red Hat OpenStack Platform 环境中运行的服务容器仍然从它们所在的主机接收时间。

第 13 章 强化基础架构和虚拟化

您可以强化物理和虚拟环境，以更好地防止内部和外部威胁。

13.1. RED HAT OPENSTACK PLATFORM 的 HARWARE

当您添加用于云环境中的硬件时，请确保支持硬件虚拟化。禁用不使用的硬件功能。

流程

1. 检查 Red Hat OpenStack 认证的硬件以确保您的硬件被支持。
2. 检查硬件虚拟化是否可用并启用：

```
cat /proc/cpuinfo | egrep "vmx|svm"
```

3. 确保您的所有固件在硬件平台上是最新的。详情请查看硬件厂商文档。

13.2. 云环境中的软件更新

为安全、性能和可支持性保持 Red Hat OpenStack Platform (RHOSP)更新。

- 如果在更新时包含内核更新，您必须重启您更新的物理系统或实例。
- 更新 OpenStack Image (glance)镜像，以确保新创建的实例具有最新的更新。
- 如果您有选择地更新 RHOSP 上的软件包，请确保包含所有安全更新。有关最新漏洞和安全更新的更多信息，请参阅：
  - RHSA-announce
  - 勘误通知
  - 安全公告

13.3. 更新 OPENSTACK 环境中的 SSH 密钥

另外，在以下情况下，您必须更新 SSH 密钥少于 2048 位：

- 在从 RHOSP 16.2 升级到 17.1 的过程中，您要将 Red Hat OpenStack Platform (RHOSP)集群升级到 RHEL 9.2
- 从 RHOSP 17.0 或 17.1 次版本更新到最新的 RHOSP 17.1 次版本

运行 **ssh_key_rotation.yaml** Ansible Playbook，以安全地自动轮转 SSH 密钥。在 overcloud 上，备份密钥存储在以下目录中：

/home/{{ ansible_user_id }}/backup_keys/{{ ansible_date_time.epoch }}/authorized_keys"

先决条件

- 您有一个完全安装的 RHOSP 环境。

流程

1. 登录到 RHOSP director：

```
ssh stack@director
```

2. 运行 ansible-playbook **ssh_key_rotation.yaml** ：

```
$  ansible-playbook \
-i /home/stack/overcloud-deploy/<stack_name>/tripleo-ansible-inventory.yaml \ 1
-e undercloud_backup_folder=/home/stack/overcloud_backup_keys \ 2
/usr/share/ansible/tripleo-playbooks/ssh_key_rotation.yaml
```

1. 将 **<stack_name>** 替换为 overcloud 堆栈的名称。
2. 在 **堆栈** 主目录中指定您选择的备份目录。

注意

如果您有单单元部署，已完成此步骤。如果您有多个单元，则必须继续。

3. 对您拥有的每个额外单元重新运行 playbook：

```
ansible-playbook \
-i /home/stack/overcloud-deploy/<stack_name>/tripleo-ansible-inventory.yaml \ 1
-e rotate_undercloud_key=false \ 2
-e ansible_ssh_private_key_file=/home/stack/overcloud_backup_keys/id_rsa 3
tripleo-ansible/playbooks/ssh_key_rotation.yaml
```

1. 将 **<stack_name>** 替换为单元的堆栈名称。每个单元都有不同的名称。
2. 您必须通过将 **rotate_undercloud_key** 参数设置为 **false** 来确保 undercloud 密钥 **不会被** 轮转。
3. 指向 SSH 备份密钥，以便在轮转旧 SSH 密钥后向其他单元中的 Compute 主机进行身份验证。

13.4. 限制硬件和软件功能

仅启用您使用的硬件和软件功能，以便减少代码受到攻击的可能性。一些功能应只在可信的环境中启用。

PCI passthrough

PCI 透传允许实例直接访问节点上的 PCI 设备。具有 PCI 设备访问的实例可能会允许恶意参与者对固件进行修改。另外，一些 PCI 设备具有直接内存访问(DMA)。当您为使用 DMA 的设备提供实例控制时，它可以获得任意物理内存访问。

您必须为特定用例启用 PCI 透传，如网络功能虚拟化(NFV)。除非部署需要，否则不要启用 PCI 透传。

内核相同的页面合并

内核相同的页面合并(KSM)是一种通过重复数据删除和共享内存页面来减少内存使用的功能。当两个或多个虚拟机在内存中具有相同的页面时，可以共享这些页面以获得更高的密度。内存重复数据删除（Deduplication）策略容易受到侧信通道攻击的影响，且只应在可信环境中使用。在 Red Hat OpenStack Platform 中，默认禁用 KSM。

13.5. RED HAT OPENSTACK PLATFORM 上的 SELINUX

Security-Enhanced Linux (SELinux)是强制访问控制(MAC)的实现。MAC 通过限制系统中允许什么进程或应用程序来限制攻击的影响。有关 SELinux 的详情，请查看 SELinux 是什么？

为 Red Hat OpenStack Platform (RHOSP)服务预配置了 SELinux 策略。在 RHOSP 上，SELinux 配置为在单独的安全上下文下运行每个 QEMU 进程。在 RHOSP 中，SELinux 策略有助于保护虚拟机监控程序主机和实例免受以下威胁：

hypervisor 威胁

在实例内运行的被破坏的应用程序可以攻击虚拟机监控程序访问底层资源。如果一个实例可以访问虚拟机监控程序操作系统，物理设备和其他应用程序可能会成为目标。这个威胁代表了显著的风险。虚拟机监控程序上的危害也可以破坏固件、其他实例和网络资源。

实例威胁

在实例中运行的已被破坏的应用程序会攻击虚拟机监控程序访问或控制另一个实例及其资源，或者实例文件镜像。用于保护实际网络的管理策略不会直接应用到虚拟环境。因为每个实例都是 SELinux 标记的进程，所以每个实例都有一个安全边界，由 Linux 内核强制。

在 RHOSP 中，磁盘上的实例镜像文件使用 SELinux 数据类型 **svirt_image_t** 标记。当实例被开机时，SELinux 会将随机数字标识符附加到镜像中。随机数字标识符可以防止被入侵的 OpenStack 实例访问其他容器。SELinux 能够为每个虚拟机监控程序节点上分配最多 524,288 个数字标识符。

13.6. 检查容器化服务

Red Hat OpenStack Platform 附带的 OpenStack 服务在容器内运行。容器化允许在不依赖项相关的冲突的情况下对服务进行开发和升级。当服务在容器中运行时，也会包含该服务的潜在漏洞。

您可以按照以下步骤获取环境中运行的服务的信息：

流程

- 使用 'podman inspect 获取信息，如绑定挂载的主机目录：
  例如：

```
$ sudo podman inspect <container_name> | less
```

  将 <container_name> 替换为容器的名称。例如，**nova 计算**。

- 检查位于 **/var/log/containers** 中的服务的日志：
  例如：

```
sudo less /var/log/containers/nova/nova-compute.log
```

- 在容器内运行交互式 CLI 会话：
  例如：

```
podman exec -it nova_compute /bin/bash
```

注意

您可以对服务进行更改，以直接在容器内进行测试。容器重启时，所有更改都会丢失。

13.7. 对容器化服务进行临时更改

您可以对容器重启时保留的容器化服务进行更改，但不会影响 Red Hat OpenStack Platform (RHOSP)集群的永久配置。这可用于测试配置更改，或者在故障排除时启用调试级别日志。您可以手动恢复更改。或者，在 RHOSP 集群上运行重新部署会将所有参数重置为其永久配置。

使用位于 **/var/lib/config-data/puppet-generated/[service]** 的配置文件对服务进行临时更改。以下示例在 nova 服务中启用调试：

流程

1. 编辑绑定到 **nova_compute** 容器的 **nova.conf** 配置文件。将 **debug** 参数的值设置为 **True** ：

```
$ sudo sed -i 's/^debug=.*/debug=True' \
/var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf
```

警告

OpenStack 文件的配置文件是 **ini** 文件，其中包括多个部分，如 **[DEFAULT]** 和 **[database]**。每个部分唯一的参数在整个文件中可能并不是唯一的。请谨慎使用 **sed**。您可以通过运行 **egrep -v "^$|^#" [configuration_file] | grep [parameter]** 以检查一个**参数**是否在配置文件中出现了多次。

2. 重启 nova 容器：

```
sudo podman restart nova_compute
```

13.8. 对容器化服务进行永久更改

您可以使用 heat 对 Red Hat OpenStack Platform (RHOSP)服务中的容器化服务进行永久更改。使用您首次部署 RHOSP 时使用的现有模板，或创建新模板以添加到部署脚本中。在以下示例中，libvirt 的私钥大小增加到 **4096**。

流程

1. 创建一个名为 **libvirt-keysize.yaml** 的新 **yaml** 模板，并使用 **LibvirtCertificateKeySize** 参数将默认值从 **2048** 增加到 **4096**。

```
cat > /home/stack/templates/libvirt-keysize.yaml
parameter_defaults:
      LibvirtCertificateKeySize: 4096
EOF
```

2. 将 **libvirt-keysize.yaml** 配置文件添加到部署脚本中：

```
openstack overcloud deploy --templates \
...
-e /home/stack/templates/libvirt-keysize.yaml
```

```
...
```

3. 重新运行部署脚本：

```
./deploy.sh
```

13.9. 固件更新

物理服务器使用复杂的固件来启用和运行服务器硬件以及远程管理卡，这些卡可以拥有自己的安全漏洞，从而可能允许系统访问和中断。为解决这些问题，硬件供应商会发布与操作系统更新分开安装的固件更新。您需要一个按常规计划检索、测试和实施这些更新的操作安全流程，请注意固件更新通常需要重新引导物理主机才能生效。

13.10. 使用 SSH 横幅文本

您可以设置横幅，将控制台消息显示给通过 SSH 连接的所有用户。您可以使用环境文件中的以下参数，将横幅文本添加到 **/etc/issue** ：考虑自定义此示例文本以满足您的要求。

```
resource_registry:
  OS::TripleO::Services::Sshd:
    /usr/share/openstack-tripleo-heat-templates/deployment/sshd/sshd-baremetal-puppet.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.*
   ******************************************************************
```

要将此更改应用到您的部署，请将设置保存为名为 **ssh_banner.yaml** 的文件，然后将其传递给 **overcloud deploy** 命令，如下所示：& **lt;full environment** > 表示您仍必须包含所有原始部署参数。例如：

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

13.11. 系统事件的审计

维护所有审计事件记录可帮助您建立系统基线、执行故障排除或分析导致特定结果的事件序列。审计系统可以记录许多类型的事件，如系统时间更改、更改为 Mandatory/Discretionary Access Control，以及创建/删除用户或组。

可以使用环境文件来创建规则，然后由 director 注入 **/etc/audit/audit.rules**。例如：

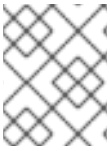
```
resource_registry:
  OS::TripleO::Services::AuditD: /usr/share/openstack-tripleo-heat-
templates/deployment/auditd/auditd-baremetal-puppet.yaml
parameter_defaults:
```

```
    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
```

13.12. 管理防火墙规则

防火墙规则在部署过程中自动应用到 overcloud 节点，并仅公开 OpenStack 正常工作所需的端口。您可以根据需要指定额外的防火墙规则。例如，要为 Zabbix 监控系统添加规则：

```
parameter_defaults:
  ControllerExtraConfig:
    ExtraFirewallRules:
      '301 allow zabbix':
      dport: 10050
      proto: tcp
      source: 10.0.0.8
```



注意

如果没有设置 **action** 参数，则结果将 **接受**。您只能将 **action** 参数设置为 **drop**、**insert** 或 **append**。

您还可以添加限制访问的规则。规则定义中使用的数字将决定规则的优先级。例如，Rabx 的规则号默认为 **109**。如果要 restrainit，请将其切换为使用较低值：

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

在本例中，**098** 和 **099** 是任意选择的数字，这些数字低于 RabbitMQ 的规则编号 **109**。要确定规则的数字，您可以检查相应节点上的 iptables 规则；对于 RabbitMQ，您要检查控制器：

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

或者，您还可以从 puppet 定义中提取端口要求。例如，RabbitMQ 的规则存储在 **puppet/services/rabbitmq.yaml** 中：

```
ExtraFirewallRules:
  '109 rabbitmq':
    dport:
      - 4369
      - 5672
      - 25672
```

可以为规则设置以下参数：

- **dport** ：与规则关联的目标端口。
- **sport** ：与规则关联的源端口。
- **proto** ：与规则关联的协议。默认为 *tcp*
- **action** ：与规则关联的操作策略。默认为 *INSERT*，并将 jump 设为 *ACCEPTS*。
- **State** ：与规则关联的状态阵列。默认为 [*NEW*]
- **source**: 与规则关联的源 IP 地址。
- **接口** ：与规则关联的网络接口。
- **链** ：与规则关联的链。默认为 *INPUT*
- **目的地** ：与规则关联的目的地 cidr。

13.13. 使用 AIDE 进行入侵检测

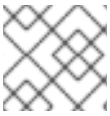
AIDE （高级入侵检测环境）是一个文件和目录完整性检查器。它用于检测未授权文件篡改或更改的事件。例如，如果系统密码文件发生更改，AIDE 可以提醒您。

AIDE 通过分析系统文件，然后编译文件哈希的完整性数据库即可工作。然后，数据库充当一个比较点，以验证文件和目录的完整性并检测更改。

director 包含 AIDE 服务，允许您向 AIDE 配置中添加条目，然后供 AIDE 服务用于创建完整性数据库。例如：

```
resource_registry:
  OS::TripleO::Services::Aide:
    /usr/share/openstack-tripleo-heat-templates/deployment/aide/aide-baremetal-ansible.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
```



注意

上例没有被主动维护或基准测试，因此您应该选择适合您的要求的 AIDE 值。

1. 声明了一个名为 **TripleORules** 的别名，以避免每次重复出现相同的属性。
2. 别名接收 **p+sha256** 属性。在 AIDE 术语中，这会被解释为：监视所有文件权限 **p**，带有完整性 checksum **sha256**。

有关 AIDE 配置文件可用属性的完整列表，请参见 的 AIDE MAN 页面（https://aide.github.io/）。

完成以下内容以将更改应用到您的部署：

1. 将设置保存为 **/home/stack/templates/** 目录中的名为 **aide.yaml** 的文件。
2. 编辑 **aide.yaml** 环境文件，使其包含适合您的环境的参数和值。
3. 在 **openstack overcloud deploy** 命令中包含 **/home/stack/templates/aide.yaml** 环境文件，以及特定于您的环境的所有其他必要的 heat 模板和环境文件：

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

13.13.1. 使用复杂 AIDE 规则

可使用前面描述的格式创建复杂的规则。例如：

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

以上指令将转换为以下指令：监控权限、索引节点、链接数、用户、组、大小、块计数、mtime、ctime、ctime，使用 sha256 生成校验和。

请注意，别名应始终具有 **1** 的顺序位置，这表示它位于 AIDE 规则的顶部，并递归应用到下面的所有值。

在别名后面是要监控的目录。请注意，可以使用正则表达式。例如，我们为 **var** 目录设置了监控，但使用了 not（**!**）语句（**'!/var/log.*'** 和 **'!/var/spool.*'**）进行了覆盖。

13.13.2. 其他 AIDE 值

以下 AIDE 值也可用：

AideConfPath: aide 配置文件的完整 POSIX 路径，默认为 **/etc/aide.conf**。如果没有要求更改文件位置，则建议使用默认路径来坚持。

AideDBPath: AIDE 完整性数据库的完整 POSIX 路径。这个值是可配置的，允许操作员声明自己的完整路径，因为通常 AIDE 数据库文件存储在只读文件挂载上。

AideDBTempPath: AIDE 完整性临时数据库的完整 POSIX 路径。AIDE 初始化新数据库时会创建此临时文件。

aideHour: 这个值是设置 hour 属性作为 AIDE cron 配置的一部分。

AideMinute: 这个值是设置 minute 属性，作为 AIDE cron 配置的一部分。

AideCronUser: 这个值是将 linux 用户设置为 AIDE cron 配置的一部分。

AideEmail: 此值设置每次进行 cron 运行时接收 AIDE 的电子邮件地址。

AideMuaPath: 这个值设置邮件用户代理的路径，用于将 AIDE 报告发送到 **AideEmail** 中设置的电子邮件地址。

13.13.3. AIDE 的 Cron 配置

AIDE director 服务允许您配置 cron 任务。默认情况下，它将报告发送到 **/var/log/audit/** ；如果您想要使用电子邮件警报，则启用 **AideEmail** 参数，以将警报发送到配置的电子邮件地址。请注意，对关键警报的依赖电子邮件可能会受到系统中断和意外消息过滤的影响。

13.13.4. 考虑系统升级的影响

执行升级时，AIDE 服务将自动重新生成一个新的完整性数据库，以确保正确重新计算所有升级的文件具有更新的校验和。

如果 **openstack overcloud deploy** 作为后续运行调用到初始部署，并且更改了 AIDE 配置规则，则 director AIDE 服务将重建数据库，以确保新配置属性封装在完整性数据库中。

13.14. REVIEW SECURETTY

securetty 允许您禁用任何控制台设备(tty)的 root 访问权限。此行为由 **/etc/securetty** 文件中的条目管理。例如：

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

parameter_defaults:
  TtyValues:
    - console
    - tty1
    - tty2
    - tty3
    - tty4
    - tty5
    - tty6
```

13.15. IDENTITY SERVICE 的 CADF 审计

全面的审计流程可帮助您检查 OpenStack 部署的持续安全状态。这对 keystone 尤为重要，因为其在安全模型中的角色。

Red Hat OpenStack Platform 已采用 Cloud Auditing Data Federation (CADF)作为审计事件的数据格式，keystone 服务为 Identity 和 Token 操作生成 CADF 事件。您可以使用 **KeystoneNotificationFormat** 为 keystone 启用 CADF 审计：

```
parameter_defaults:
  KeystoneNotificationFormat: cadf
```

13.16. 查看 LOGIN.DEFS 值

要对新系统用户（非keystone）强制实施密码要求，director 可以通过以下示例参数向 **/etc/login.defs** 添加条目：

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

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

第 14 章 强化仪表板服务

Dashboard 服务(horizon)为用户提供自助服务门户，用于在管理员设置的限制内置备自己的资源。使用与 OpenStack API 相同的敏感度管理 Dashboard 服务的安全性。

14.1. 调试仪表板服务

DEBUG 参数的默认值为 **False**。在生产环境中保留默认值。仅在调查期间更改此设置。当您将 **DEBUG** 参数的值更改为 **True** 时，Djan 可以输出 stack straces 到包含敏感 Web 服务器状态信息的浏览器用户。

当 **DEBUG** 参数的值为 **True** 时，**ALLOWED_HOSTS** 设置也会禁用。有关配置 **ALLOWED_HOSTS** 的更多信息，请参阅配置 ALLOWED_HOSTS。

14.2. 选择域名

最佳实践是将 Dashboard 服务(horizon)部署到第二个级别域，而不是在任何级别上共享域。以下是每个示例：

- 第二级域： **https://example.com**
- 共享子域 ：**https://example.public-url.com**

根据浏览器的 **same-origin** 策略，将 Dashboard 服务部署到专用第二个级别域，将 Cookie 和安全令牌与其他域隔离。当部署到子域中时，Dashboard 服务的安全性等同于部署在同一第二级域中的最低安全应用程序。

您可以通过避免由 Cookie 支持的会话存储并配置 HTTP Strict Transport Security (HSTS) （本指南中描述）来进一步缓解这个风险。

注意

不支持在安全域上部署 Dashboard 服务，如 **https://example/**。

14.3. 配置 ALLOWED_HOSTS

Horizon 基于 python Django Web 框架构建，这需要防止与误导 HTTP 主机标头关联的安全威胁。要应用此保护，请将 **ALLOWED_HOSTS** 设置配置为使用 OpenStack 仪表板提供的 FQDN。

当您配置 **ALLOWED_HOSTS** 设置时，任何带有与此列表中值不匹配的 HTTP 请求都会被拒绝，并引发错误。

流程

1. 在模板中的 **parameter_defaults** 下，设置 **HorizonAllowedHosts** 参数的值：

```
parameter_defaults:
  HorizonAllowedHosts: <value>
```

   将 **<value>** 替换为 OpenStack 仪表板提供的 FQDN。

2. 使用修改后的模板部署 overcloud，以及您的环境所需的所有其他模板。

14.4. 跨站点脚本(XSS)

OpenStack 控制面板在大多数字段中接受整个 Unicode 字符集。恶意攻击者可能会试图使用此可扩展性来测试跨站点脚本(XSS)漏洞。OpenStack Dashboard 服务(horizon)具有强化针对 XSS vulnerabilites 的工具。在自定义仪表板中，务必要确保正确使用这些工具。当您对自定义仪表板执行审计时，请注意以下几点：

- **mark_safe** 函数。
- **is_safe** - 与自定义模板标签一起使用。
- **safe** 模板标签。
- 任何自动转义都会关闭，任何 JavaScript 可能会评估不正确的转义数据。

14.5. 跨站点请求 FORGERY (CSRF)

使用多个 JavaScript 实例的仪表板应该会根据使用 **@csrf_exempt** decorator 等漏洞进行审核。在降低 CORS (Cross Origin Resource Sharing)限制前，评估没有遵循推荐的安全设置的仪表板。将您的 Web 服务器配置为发送带有每个响应的严格的 CORS 标头。只允许仪表板域和协议，例如：**Access-Control-Allow-Origin: https://example.com/**。您不应该允许通配符源。

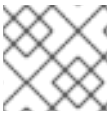
14.6. 允许嵌入的 IFRAME

DISALLOW_IFRAME_EMBED 设置不允许仪表板嵌入到 iframe 中。传统浏览器仍然可以受到跨帧脚本(XFS)漏洞的影响，因此此选项为不需要的部署添加额外的安全强化。默认设置为 **True**，但如果需要，可以使用环境文件来禁用它。

流程

- 您可以使用以下参数来允许 iframe 嵌入：

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



注意

只有当潜在安全影响被完全理解后，这些设置才应设置为 **False**。

14.7. 为 DASHBOARD 流量使用 HTTPS 加密

建议您使用 HTTPS 加密 Dashboard 流量。您可以将它配置为使用来自认可的证书认证机构(CA)的有效可信证书。只有在所有用户浏览器中预安装信任根时，私有机构发布的证书才适合。

配置对仪表板域的 HTTP 请求，以重定向到完全限定的 HTTPS URL。

如需更多信息，请参阅 *第 7 章 在 overcloud 公共端点中启用 SSL/TLS*。

14.8. HTTP 严格传输安全性(HSTS)

HTTP 严格传输安全性(HSTS)可防止浏览器在最初进行安全连接后进行后续的不安全连接。如果您在一个公共或不信任的区中部署了 HTTP 服务，则 HSTS 尤为重要。

对于基于 director 的部署，此设置在 **/usr/share/openstack-tripleo-heat-templates/deployment/horizon/horizon-container-puppet.yaml** 文件中默认启用：

```
horizon::enable_secure_proxy_ssl_header: true
```

验证

部署 overcloud 后，检查 **local_settings** 文件是否有 Red Hat OpenStack Dashboard (horizon)进行验证。

1. 使用 **ssh** 连接到控制器：

```
$ ssh tripleo-admin@controller-0
```

2. 检查 **SECURE_PROXY_SSL_HEADER** 参数的值为 **('HTTP_X_FORWARDED_PROTO', 'https')** ：

```
sudo egrep ^SECURE_PROXY_SSL_HEADER /var/lib/config-data/puppet-generated/horizon/etc/openstack-dashboard/local_settings
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
```

14.9. 前端缓存

不建议将前端缓存工具与控制面板搭配使用，因为它会呈现直接从 OpenStack API 请求生成的动态内容。因此，**varnish** 等前端缓存层可以防止显示正确的内容。控制面板使用 Django，它直接从 Web 服务提供静态介质，并且已经从 Web 主机缓存中受益。

14.10. 会话后端

对于基于 director 的部署，horizon 的默认会话后端是 **django.contrib.sessions.backends.cache**，它与 memcached 结合使用。由于性能的原因，这种方法首选本地内存缓存，对于高可用性和负载平衡安装是安全的，而且能够跨多个服务器共享缓存，同时仍将其视为单一缓存。

您可以在 director 的 **horizon.yaml** 文件中查看这些设置：

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

14.11. 查看 SECRET 密钥

控制面板取决于某些安全功能的共享 **SECRET_KEY** 设置。secret 密钥应该是随机生成的字符串，至少为 64 个字符，必须在所有活跃的仪表板实例间共享。这个密钥的危害可能会允许远程攻击者执行任意代码。轮转此密钥会导致现有用户会话和缓存无效。不要将此密钥提交到公共存储库。

对于 director 部署，此设置作为 **HorizonSecret** 值进行管理。

14.12. 配置会话 COOKIE

可以打开仪表板会话 Cookie 以根据浏览器技术（如 JavaScript）进行交互。对于随处使用 TLS 的 director 部署，您可以使用 **HorizonSecureCookies** 设置来强化此行为。

注意

切勿将 CSRF 或会话 Cookie 配置为使用带前点的通配符域。

14.13. 验证密码复杂性

OpenStack Dashboard (horizon)可以使用密码验证检查来强制实施密码复杂性。

流程

1. 为密码验证指定一个正则表达式，并为失败的测试显示帮助文本。以下示例要求用户以长度为 8 到 18 个字符创建密码：

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

1. 将此更改应用到您的部署。将设置保存为名为 **horizon_password.yaml** 的文件，然后将其传递给 **overcloud 部署命令**，如下所示：& **lt;full environment** > 表示您仍必须包含所有原始部署参数。例如：

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

14.14. 强制管理员密码检查

默认情况下，以下设置设置为 **True**，但如果需要，可以使用环境文件来禁用它。

注意

只有当潜在安全影响被完全理解后，这些设置才应设置为 **False**。

流程

Dashboard 的 **local_settings.py** 文件中的 **ENFORCE_PASSWORD_CHECK** 设置在 *Change Password* 表单中显示 *Admin Password* 字段，这有助于验证管理员是否启动密码更改。

- 您可以使用环境文件禁用 **ENFORCE_PASSWORD_CHECK** ：

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

14.15. 禁用密码显示

disable_password_reveal 参数默认设置为 **True**，但如果需要，可以使用环境文件来禁用它。密码显示按钮允许用户在仪表板中查看他们要输入的密码。

流程

- 在 **ControllerExtraConfig** 参数下，包含 **horizon::disable_password_reveal: false**。将它保存到 heat 环境文件中，并将其包含在您的部署命令中。

Example

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

注意

只有当潜在安全影响被完全理解后，这些设置才应设置为 **False**。

14.16. 显示仪表板的登录横幅

HIPAA、PCI-DSS 和美国政府等监管行业需要您显示用户登录横幅。Red Hat OpenStack Platform (RHOSP) 仪表板 (horizon) 使用默认的主题 (RCUE)，它存储在 horizon 容器中。

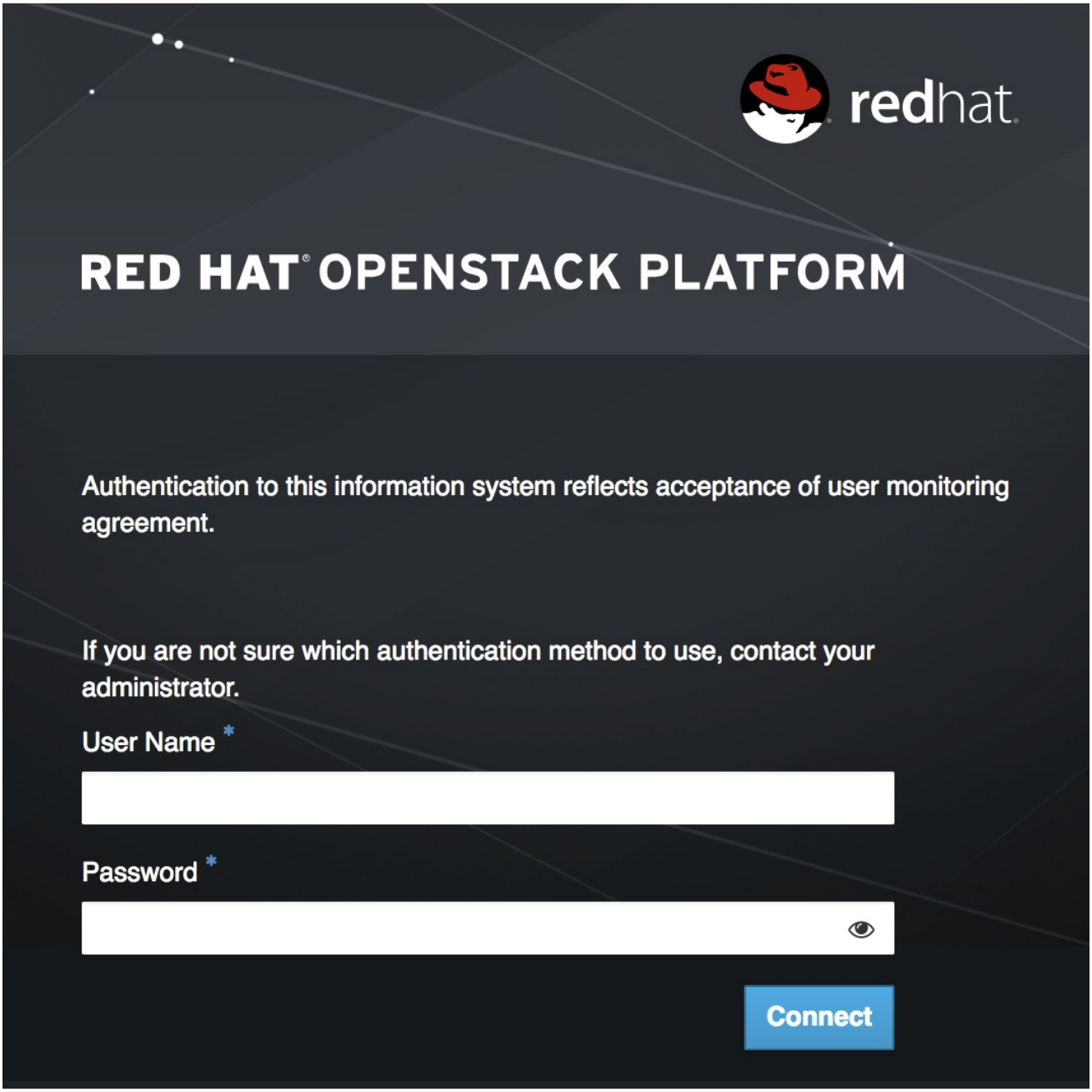
在自定义 Dashboard 容器中，您可以通过手动编辑 **/usr/share/openstack-dashboard/openstack_dashboard/themes/rcue/templates/auth/login.html** 文件来创建日志横幅：

流程

1. 在 **{% include 'auth/_login.html' %}** 部分前输入所需的 logon banner。允许 HTML 标签：

```
<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>
```

上面的示例生成类似如下的仪表板：



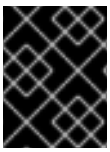
其他资源

- 自定义仪表板

14.17. 限制文件上传的大小

您可以选择配置仪表板来限制文件上传的大小；此设置可能是各种安全强化策略的要求。

LimitRequestBody - 此值（以字节为单位）限制了您可以使用控制面板传输的文件的最大大小，如镜像和其他大型文件。



重要

这个设置还没有被红帽正式测试。在将此设置部署到生产环境中之前，建议您全面测试此设置的影响。

注意

如果值太小，文件上传将失败。

例如，此设置将每个文件上传限制为最多 10 GB (**10737418240**)。您需要调整此值以符合您的部署。

- **/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>
```

注意

这些配置文件由 Puppet 管理，因此每当运行 **openstack overcloud 部署** 进程时，任何未管理的更改都会被覆盖。

第 15 章 强化网络服务

Networking 服务(neutron)是 Red Hat OpenStack Platform (RHOSP)的软件定义型网络(SDN)组件。RHOSP 网络服务管理进出虚拟机实例的内部和外部流量，并提供路由、分段、DHCP 和元数据等核心服务。它为虚拟网络功能提供 API，并管理交换机、路由器、端口和防火墙。

有关 Red Hat OpenStack Platfrom 网络服务的更多信息，请参阅配置 Red Hat OpenStack Platform 网络。

本节将讨论 OpenStack 网络配置良好实践，因为它们适用于 OpenStack 部署中的项目网络安全性。

15.1. 项目网络服务工作流

OpenStack 网络为用户提供网络资源的自助服务配置。云架构师和操作员务必要评估其设计用例，从而为用户提供创建、更新和销毁可用的网络资源。

15.2. 网络资源策略引擎

OpenStack 网络内的策略引擎及其配置文件(**policy.json**)提供了在项目联网方法和对象上为用户提供精细授权的方法。OpenStack 网络策略定义会影响网络可用性、网络安全性和整体 OpenStack 安全性。云架构师和操作员应仔细评估其对用户和项目管理网络资源的访问的策略。

注意

检查默认网络资源策略非常重要，因为此策略可以被修改以符合您的安全状况。

如果您的 OpenStack 部署提供多个外部访问点到不同的安全区，请务必限制项目将多个 vNIC 附加到多个外部接入点的能力，这会桥接这些安全区，并可能导致无法预见的安全破坏。您可以使用 Compute 提供的主机聚合功能，或者将项目实例拆分为具有不同虚拟网络配置的多个项目中，以帮助缓解此风险。如需有关主机聚合的更多信息，请参阅 创建和管理主机聚合。

15.3. 安全组

安全组是安全组规则的集合。通过安全组及其规则，管理员和项目能够指定允许通过虚拟接口端口的流量和方向(ingress/egress)的类型。在 OpenStack 网络中创建虚拟接口端口时，它会与安全组关联。规则可以添加到默认安全组中，以便根据每个部署方式更改行为。

在使用 Compute API 修改安全组时，更新的安全组将应用到实例上的所有虚拟接口端口。这是因为 Compute 安全组 API 是基于实例的，而不是基于端口的，如 neutron 中找到。

15.4. 缓解 ARP 欺骗

OpenStack 网络具有内置功能，可帮助缓解对实例的 ARP 欺骗。除非仔细考虑导致的风险，否则不应禁用。

15.5. 使用安全协议进行身份验证

在 **/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf** 中检查 **[keystone_authtoken]** 部分下的 **auth_uri** 的值是否已设置为以 'https 开头的 Identity API 端点：

第 16 章 在 RED HAT OPENSTACK PLATFORM 上强化块存储

OpenStack Block Storage (cinder)是一种提供软件（服务和库）的服务，为自助服务管理永久块级存储设备。这会按需访问块存储资源，以用于计算(nova)实例。这通过将块存储池虚拟化到各种后端存储设备（可以是软件实施或传统硬件存储产品），从而创建软件定义的存储。这的主要功能是管理块设备的创建、附加和分离。消费者不需要了解后端存储设备的类型或位置。

计算实例使用行业标准存储协议（如 iSCSI、ATA 以太网或光纤通道）存储和检索块存储。这些资源使用 OpenStack 原生标准 HTTP RESTful API 管理和配置。

16.1. 为请求的正文设置最大大小

如果没有定义每个请求的最大正文大小，攻击者可以制作大量大小的任意 OSAPI 请求，从而导致服务崩溃，最后导致服务遭受服务攻击。分配 t he 最大值可确保任何恶意的恶意请求都被阻止，以确保服务的持续可用性。

检查 **cinder.conf** 中的 **[oslo_middleware]** 部分下的 **max_request_body_size** 是否已设置为 **114688**。

16.2. 启用卷加密

未加密的卷数据使卷托管平台成为攻击者的高价值目标，因为它允许攻击者读取许多不同的虚拟机的数据。另外，物理存储介质可以被 stolen、重新挂载并从其他机器访问。加密卷数据和卷备份可帮助减少这些风险，并提供对卷托管平台的深度。块存储(cinder)可以在写入磁盘前加密卷数据，因此请考虑启用卷加密，并使用 Barbican 进行私钥存储。

16.3. VOLUME WIPING

可以通过多种方式擦除块存储设备。传统的方法是将 **lvm_type** 设置为 thin，然后使用 **volume_clear** 参数。或者，如果使用卷加密功能，则在删除卷加密密钥时不需要卷 wiping。

注意

在以前的版本中，**lvm_type=default** 用于表示擦除。虽然此方法仍有效，但不建议通过 **lvm_type=default** 设置安全删除。

volume_clear 参数可以接受 **zero** 或 **shred** 作为参数。**零** 将写入一个零到设备。**shred** 操作将编写预定的位模式的三个通过。

第 17 章 强化共享文件系统(MANILA)

共享文件系统服务(manila)提供一组服务，用于在多项目云环境中管理共享文件系统。通过 manila，您可以创建一个共享文件系统并管理其属性，如可见性、可访问性和配额。

有关共享文件系统服务(manila)的更多信息，请参阅 配置持久性存储 指南。

17.1. MANILA 的安全注意事项

Manila 通过 keystone 注册，允许您使用 **manila 端点** 命令查找 API。例如：

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

默认情况下，manila API 服务只侦听端口 **8786** 中的 **tcp6**，它支持 IPv4 和 IPv6。

Manila 使用多个配置文件；这些文件存储在 **/var/lib/config-data/puppet-generated/manila/** 中：

```
api-paste.ini
manila.conf
policy.json
rootwrap.conf
rootwrap.d

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

建议您将 manila 配置为在非 root 服务帐户下运行，并更改文件权限，以便只有系统管理员才能修改它们。Manila 期望只有管理员才能写入配置文件，服务只能通过 **manila** 组中的组成员资格读取它们。其他用户不得读取这些文件，因为它们包含服务帐户密码。

注意

只有 root 用户应该可以写入 **rootwrap.conf** 中的 **manila-rootwrap** 配置，**manila-rootwrap** 命令会过滤 **rootwrap.d/share.filters** 中的共享节点。

17.2. MANILA 的网络和安全模型

manila 中的共享驱动程序是一个 Python 类，可以为后端设置来管理共享操作，其中一些特定于供应商的。后端是 **manila-share** 服务的实例。Manila 具有许多不同存储系统的共享驱动程序，同时支持商业供应商和开源解决方案。每个共享驱动都支持一个或多个后端模式： **共享服务器**，**无共享服务器**。管理员通过在 **manila.conf** 中指定模式（使用 **driver_handles_share_servers** ）来选择模式。

共享服务器是导出共享文件系统的逻辑网络附加存储(NAS)服务器。当今的后端存储系统非常复杂，可以隔离不同 OpenStack 项目之间的数据路径和网络路径。

由 manila 共享驱动程序置备的共享服务器将在属于项目用户的隔离网络上创建它。**共享** 服务器模式可以使用扁平网络或网段的网络进行配置，具体取决于网络提供程序。

对于不同的模式，可以有单独的驱动程序来使用相同的硬件。根据所选的模式，您可能需要通过配置文件提供更多配置详情。

17.3. 共享后端模式

每个共享驱动程序至少支持其中一个可用驱动程序模式：

- 共享服务器 - **driver_handles_share_servers = True** - 共享驱动程序创建共享服务器并管理共享服务器生命周期。
- 无共享服务器 - **driver_handles_share_servers = False** - 管理员（而不是共享驱动程序）使用网络接口管理裸机存储，而不依赖于共享服务器的存在。

无共享服务器模式 - 在这个模式中，驱动程序不会设置共享服务器，因此不需要设置任何新的网络接口。假设存储控制器由驱动程序管理，具有需要的所有网络接口。驱动程序在没有之前创建共享服务器的情况下直接创建共享。要使用此模式中操作的驱动程序创建共享，manila 不需要用户创建任何私有共享网络。

注意

在没有共享服务器模式 中，manila 将假定所有共享都可通过所有项目访问的网络接口。

在没有共享服务器 模式中，共享驱动程序无法处理共享服务器生命周期。管理员应该处理可能需要提供项目隔离所需的存储、网络和其他主机配置。在这个模式中，管理员可以将存储设置为导出共享的主机。OpenStack 云中的所有项目共享一个共同的网络管道。缺少隔离可能会影响安全性和服务质量。当使用不处理共享服务器的共享驱动程序时，云用户不能确保不受信任的用户不能在文件系统的顶级目录中访问其共享。在公有云中，所有网络带宽都可以被一个客户端使用，因此管理员应谨慎这样做。网络平衡可以通过任何方式实现，而不一定只是使用 OpenStack 工具实现。

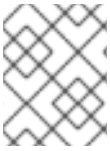
共享服务器模式 - 在这个模式中，驱动程序能够创建共享服务器并将其插入到现有的 OpenStack 网络。Manila 确定需要新的共享服务器，并提供共享驱动程序所需的所有网络信息，以创建必要的共享服务器。

在处理共享服务器的驱动程序模式中创建共享时，用户必须提供一个共享网络，他们希望导出其共享。Manila 使用此网络为此网络上的共享服务器创建网络端口。

用户可以在 **share servers** 和 **no share servers** 后端模式中配置 **security services**。但是，**如果没有共享服务器** 后端模式，管理员必须在主机上手动设置所需的身份验证服务。在 **共享服务器** 模式中，manila 可以在其生成的共享服务器上配置用户标识的安全服务。

17.4. MANILA 的网络要求

Manila 可以与不同的网络类型集成： **扁平**、**GRE**、**VLAN**、**VXLAN**。



注意

Manila 仅存储数据库中的网络信息，其真正的网络由网络提供程序提供。Manila 支持使用 OpenStack Networking 服务(neutron)和 "standalone" 预配置网络。

在 **共享服务器** 后端模式中，共享驱动程序会为每个共享网络创建和管理共享服务器。这个模式可以分为两种变化：

- **共享服务器** 后端模式的扁平网络
- **共享服务器** 后端模式中的分段网络

用户可以使用 OpenStack Networking (neutron)服务的网络和子网来创建共享网络。如果管理员决定使用 **StandAloneNetworkPlugin**，用户不需要提供任何网络信息，因为管理员在配置文件中进行了预配置。

注意

共享由某些共享驱动程序生成的服务器是利用计算服务创建的计算服务器。其中一些驱动程序不支持网络插件。

创建共享网络后，manila 会检索由网络供应商决定的网络信息：网络类型、分段标识符（如果网络使用分段）和 CIDR 标记中的 IP 块（如果网络使用分段），以及从中分配网络的 CIDR 标记中的 IP 块。

用户可以创建指定安全要求的安全服务，如 AD 或 LDAP 域或 Kerberos 域。Manila 假设在安全服务中引用的任何主机都可以从创建共享服务器的子网访问，这限制了使用此模式的情况数量。

注意

有些共享驱动程序可能不支持所有类型的分段，请参阅您使用的驱动程序规格。

17.5. 使用 MANILA 的安全服务

Manila 可以通过与网络身份验证协议集成来限制对文件共享的访问。每个项目可以拥有自己的身份验证域，与云 Keystone 身份验证域分开的功能。此项目域可用于向 OpenStack 云中运行的应用程序（包括 manila）提供授权(AuthZ)服务。可用的身份验证协议包括 LDAP、Kerberos 和 Microsoft Active Directory 身份验证服务。

17.6. 安全服务简介

在创建了共享并获取其导出位置后，用户没有挂载它的权限，并且操作文件。用户需要明确授予新共享的访问权限。

客户端身份验证和授权(authN/authZ)可以与安全服务结合使用。Manila 可以使用 LDAP、Kerberos 或 Microsoft Active 目录（如果共享驱动程序和后端支持）。

注意

在某些情况下，需要明确指定其中一个安全服务，例如 NetApp、EMC 和 Windows 驱动程序需要 Active Directory 来创建与 CIFS 协议的共享。

17.7. 安全服务管理

安全服务 是一个 manila 实体，它抽象一组选项来为特定共享文件系统协议定义安全区，如 Active Directory 域或 Kerberos 域。安全服务包含 manila 创建加入给定域的服务器所需的所有信息。

使用 API，用户可以创建、更新、查看和删除安全服务。安全服务由以下假设设计：

- 项目提供安全服务的详细信息。
- 管理员关注安全服务：它们配置此类安全服务的服务器端。
- 在 manila API 中，**security_service** 与 **share_networks** 关联。
- 共享驱动程序使用安全服务中的数据来配置新创建的共享服务器。

在创建安全服务时，您可以选择以下身份验证服务之一：

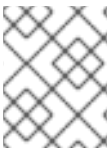
- **LDAP** - 轻量级目录访问协议。用于通过 IP 网络访问和维护分布式目录信息服务的应用程序协议。
- **Kerberos** - 网络身份验证协议，以票据为基础，允许节点通过非安全网络进行通信，以安全的方式证明其身份。
- **Active Directory** - Microsoft 为 Windows 域网络开发的目录服务。使用 LDAP、Microsoft 的 Kerberos 版本和 DNS。

Manila 允许您使用这些选项配置安全服务：

- 在项目网络中使用的 DNS IP 地址。
- 安全服务的 IP 地址或主机名。
- 安全服务的域。
- 项目使用的用户或组名称。
- 指定用户的密码，如果您指定了用户名。

现有的安全服务实体可以与共享网络实体关联，该实体告知 manila 一组共享的安全性和网络配置。您还可以查看指定共享网络的所有安全服务列表，并将它们从共享网络中解除关联。

管理员和用户作为共享所有者可以通过 IP 地址、用户、组或 TLS 证书创建带有身份验证的访问规则来管理对共享的访问。身份验证方法取决于您配置和使用的共享驱动程序和安全服务。然后，您可以将后端配置为使用特定的身份验证服务，该服务可以在没有 manila 和 keystone 的客户端中运行。



注意

不同共享驱动程序支持不同的身份验证服务。有关不同驱动程序支持功能的详情，请参考 https://docs.openstack.org/manila/latest/admin/share_back_ends_feature_support_mappi

对驱动程序的特定身份验证服务的支持并不意味着可以使用任何共享文件系统协议进行配置。支持的共享文件系统协议有 NFS、CEPHFS、CIFS、GlusterFS 和 HDFS。有关特定驱动程序及其安全服务配置的信息，请参阅驱动程序厂商的文档。

有些驱动支持安全服务，而其他驱动则不支持上述任何安全服务。例如，带有 NFS 或 CIFS 共享文件系统协议的通用驱动程序只支持通过 IP 地址进行身份验证的方法。

注意

在大多数情况下，支持 CIFS 共享文件系统协议的驱动程序可以配置为使用 Active Directory 并通过用户身份验证管理访问权限。

- 支持 GlusterFS 协议的驱动程序可用于使用 TLS 证书进行身份验证。
- 使用支持使用 IP 地址进行 NFS 协议身份验证的驱动程序是唯一支持的选项。
- 由于 HDFS 共享文件系统协议使用 NFS 访问，因此也可以将其配置为使用 IP 地址进行身份验证。

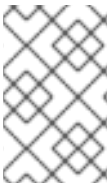
生产 manila 部署的推荐配置是使用 CIFS 共享协议创建共享并添加到其 Microsoft Active Directory 目录服务中。使用这个配置，您将获取集中数据库，以及集成 Kerberos 和 LDAP 方法的服务。

17.8. 共享访问控制

用户可以指定哪些特定客户端有权访问其创建的共享。由于 keystone 服务，由各个用户创建的共享仅对同一项目中的自身和其他用户可见。Manila 允许用户创建"公共"可见的共享。这些共享在属于其他 OpenStack 项目的用户的仪表板中可见，如果所有者授予他们访问权限，它们也可能能够挂载这些共享（如果可以在网络上访问）。

在创建共享时，使用 key **--public** 使您的共享公共项目在共享列表中查看并查看其详细信息。

根据 **policy.json** 文件，管理员和用户作为共享所有者可以通过创建访问规则来管理对共享的访问。使用 **manila access-allow**、**manila access-deny** 和 **manila access-list** 命令，您可以相应地授予、拒绝和列出对指定共享的访问权限。



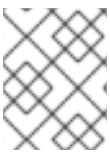
注意

Manila 不提供存储系统的端到端管理。您仍然需要单独保护后端系统不受未授权访问的影响。因此，如果某人破坏后端存储设备，则 manila API 提供的保护仍可以被绕过，从而获得带外访问。

仅创建共享时，没有与之关联的默认访问规则，并且权限可挂载它。这可以在挂载配置中用于导出协议。例如，存储上有一个 NFS 命令 **exportfs** 或 **/etc/exports** 文件，该文件控制每个远程共享并定义可以访问它的主机。如果 nobody 可以挂载共享，则为空。对于远程 CIFS 服务器，有 **net conf list** 命令显示配置。**hosts deny** 参数应由共享驱动程序设置为 **0.0.0.0/0**，这意味着任何主机都被拒绝来挂载共享。

使用 manila，您可以通过指定这些支持的共享访问级别之一来授予或拒绝对共享的访问：

- **rw** - 读写(RW)访问。这是默认值。
- **ro**- 只读(RO)访问。



注意

当管理员为某些编辑器或贡献者授予某些编辑器或贡献者的读和写(RW)访问权限，并且为其余用户(viewers)提供只读(RO)访问权限时，对公共共享的访问级别会很有用。

您还必须指定这些支持的身份验证方法之一：

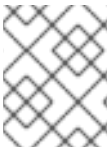
- **IP** - 使用 IP 地址来验证实例。可以通过以 CIDR 表示法以正确的 IPv4 或 IPv6 地址为客户端提供 IP 访问。

- **cert** - 使用 TLS 证书来验证实例。将 TLS 身份指定为 **IDENTKEY**。有效值是证书的通用名称(CN)中最多 64 个字符的任何字符串。
- **user** - 由指定用户或组名称进行授权。有效值是字母数字字符，可以包含一些特殊字符，且从 4 到 32 个字符。

注意

支持的身份验证方法取决于您使用的共享驱动程序、安全服务和共享文件系统协议。支持的共享文件系统协议有 MapRFS、CEPHFS、NFS、CIFS、GlusterFS 和 HDFS。支持的安全服务包括 LDAP、Kerberos 协议或 Microsoft Active Directory 服务。

要验证是否为共享正确配置了访问规则(ACL)，您可以列出其权限。



注意

为共享选择安全服务时，您需要考虑共享驱动程序是否能够使用可用的身份验证方法创建访问规则。支持的安全服务包括 LDAP、Kerberos 和 Microsoft Active Directory。

17.9. 共享类型访问控制

共享类型是一个由管理员定义的*服务类型（type of service）*，它包括一个项目可见的描述信息，以及一个项目不可见的、称为*额外规格（extra specifications）*的键值对列表。**manila-scheduler** 使用额外的规格来做出调度决策，驱动程序则控制共享创建。

管理员可以创建和删除共享类型，也可以管理额外的规格，以便在 manila 内提供含义。项目可以列出共享类型，并可使用它们创建新的共享。共享类型可以被创建为 **public** 和 **private**。这是定义其他项目是否可以在共享类型列表中看到的共享类型的可见性级别，并使用它来创建新共享。

默认情况下，共享类型创建为 public。在创建共享类型时，请使用 **--is_public** 参数设置为 **False** 以使您的共享类型私有，这将阻止其他项目在共享类型列表中看到并创建新的共享。另一方面，云中的每个项目都可以使用 **公共** 共享类型。

Manila 允许管理员授予或拒绝对项目的 **私有** 共享类型的访问权限。您还可以获取有关指定私有共享类型访问的信息。

注意

由于由于其额外规格而共享类型有助于在用户创建共享之前过滤或选择后端，因此您可以使用对共享类型的访问权限，以限制特定后端选择的客户端。

例如，**admin** 项目中的管理员可以创建名为 **my_type** 的专用共享类型，并在列表中看到它。在以下控制台示例中，省略了登录和注销，并提供了环境变量来显示当前登录的用户。

```
$ 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 项目中的 **demo** 用户可以列出类型，但名为 **my_type** 的私有共享类型对他不可见。

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

管理员可以授予 demo 项目的私有共享类型的访问权限，其 ID 等于 **df29a37db5ae48d19b349fe947fada46**:

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

因此，**demo** 项目中的用户可以看到私有共享类型，并在创建共享时使用它：

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

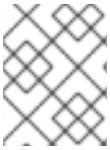
要拒绝对指定项目的访问，请使用 **manila type-access-remove <share_type> <project_id>**。

注意

例如，演示共享类型的目的，请考虑一个您有两个后端的情况：LVM 作为公共存储，Ceph 作为私有存储。在这种情况下，您可以使用 **用户/组 身份**验证方法授予对特定项目的访问权限并控制访问权限。

17.10. 策略（POLICY）

共享文件系统服务 API 使用基于角色的访问控制策略进行授权。这些策略决定了哪些用户可以以某种方式访问某些 API，并在服务的 **policy.json** 文件中定义。



注意

配置文件 **policy.json** 可以在任何位置放置。默认情况下，预期为 **/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 路径。

每当向 manila 发出 API 调用时，策略引擎使用适当的策略定义来确定是否可以接受调用。策略规则决定允许 API 调用的情况。**/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 文件的规则始终被允许，当规则是空字符串： **""** 时，基于用户角色或规则的规则；带有布尔值表达式的规则。以下是 manila 的 **policy.json** 文件的片段。预计可以在 OpenStack 版本之间有所变化。

```
{
    "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": "",
}
```

用户必须分配给您在您的策略中引用的组和角色。在使用用户管理命令时，该服务会自动执行此操作。

注意

对 **/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 的任何更改都会立即生效，允许在 manila 运行时实施新策略。手动修改策略可能会具有意外的副作用，我们不建议这样做。Manila 不提供默认策略文件；所有默认策略都在代码库中。您可以执行： **oslopolicy-sample-generator --config-file=var/lib/config-data/puppet-generated/manila/etc/manila/manila-policy-generator.conf**从 manila 代码生成默认策略

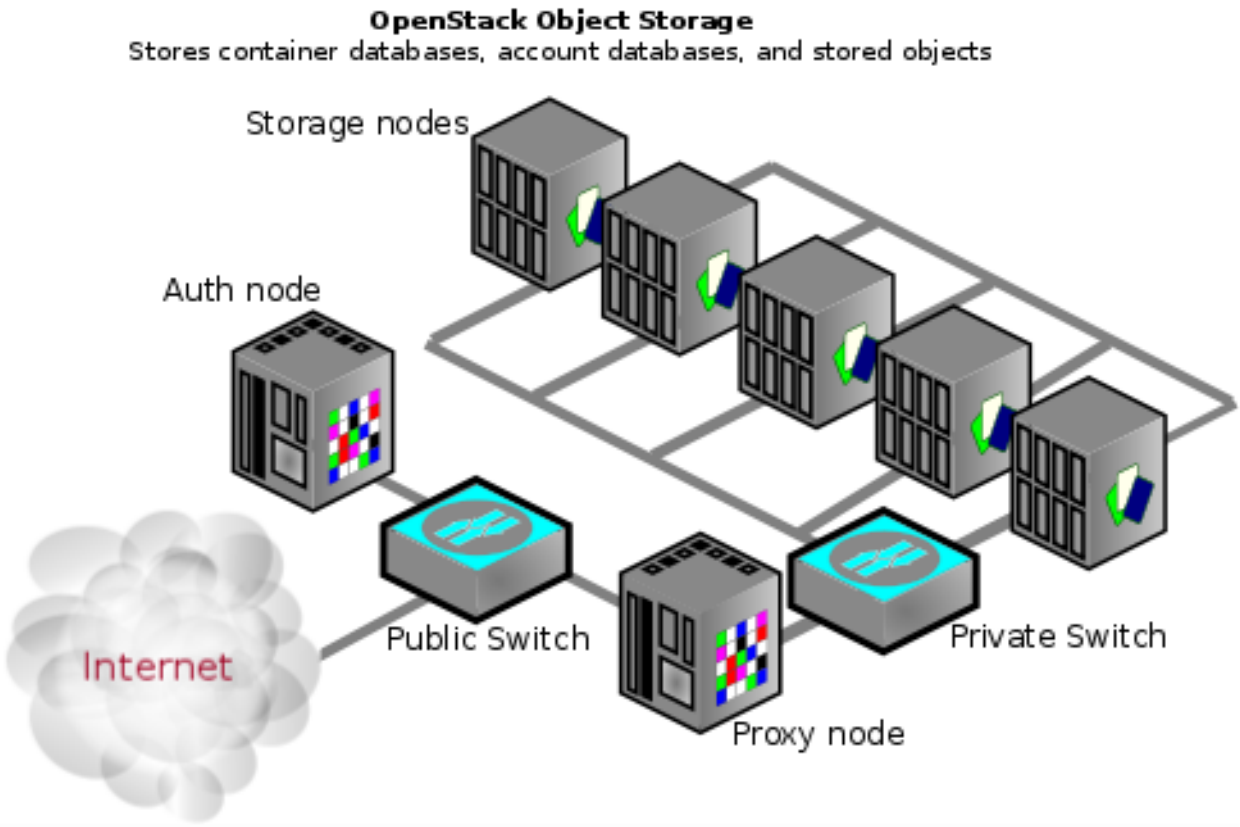
第 18 章 对象存储

Object Storage (swift)服务通过 HTTP 存储和检索数据。对象（覆盖数据）存储在组织层次结构中，可以被配置为提供匿名只读访问、ACL 定义的访问权限，甚至临时访问。Swift 支持多种通过中间件实施的基于令牌的验证机制。

应用程序使用行业标准的 HTTP RESTful API 在对象存储中存储和检索数据。后端 swift 组件遵循相同的 RESTful 模型，但一些 API （如管理持久性）会保持在集群中私有。

swift 组件属于以下主要组：

- 代理服务
- 身份验证服务
- 存储服务
  - 帐户服务
  - 容器服务
  - 对象服务



注意

对象存储安装不必面向互联网，也可能是带有机构内部网络基础架构一部分的公共交换机的私有云。

18.1. 网络安全性

swift 的安全强化从保护网络组件开始。如需更多信息，请参阅网络章节。

为获得高可用性，rsync 协议用于在存储服务节点之间复制数据。另外，当转发客户端端点和云环境之间的数据时，代理服务与存储服务通信。

注意

Swift 不使用加密或与节点间通信进行身份验证。这是因为 swift 使用原生 rsync 协议来提高性能，且不会对 rsync 通信使用 SSH。因此，您在架构图中看到私有交换机或专用网络([V]LAN)。此数据区域也应与其他 OpenStack 数据网络分开。

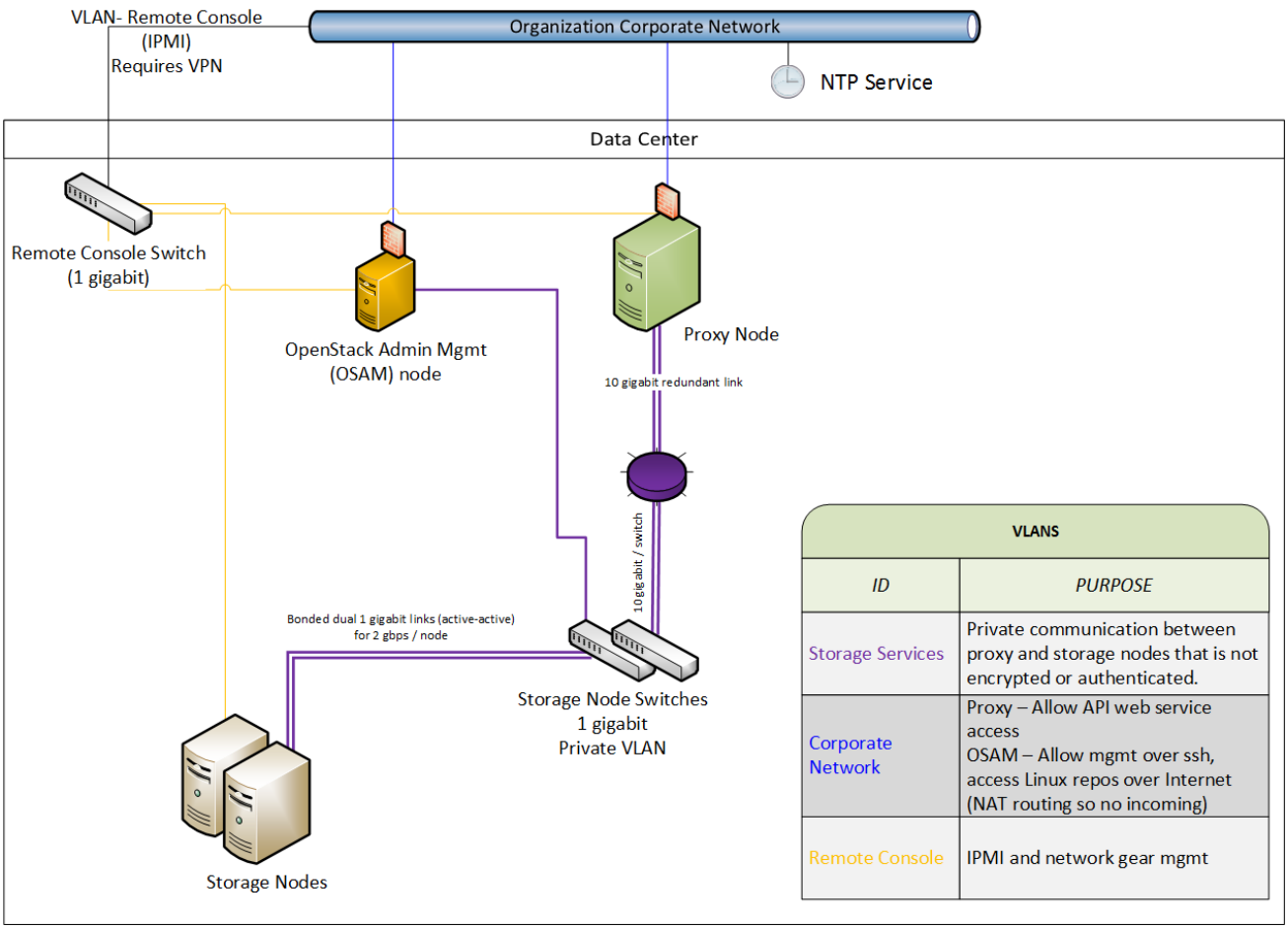
注意

将私有(V) LAN 网络段用于 data 区域中的存储节点。

这要求代理节点具有双接口（物理或虚拟）：

- 一个接口作为一个公共接口，供消费者访问。
- 另一个接口作为私有接口，可访问存储节点。

下图显示了一种可能的网络架构，将对象存储网络架构与管理节点(OSAM)搭配使用：



18.2. 以非 ROOT 用户身份运行服务

建议您将 swift 配置为在非 root (**UID 0**)服务帐户下运行。一个建议是，director 部署的用户名 **swift** 并带有一个主组 **swift**。对象存储服务包括 **proxy-server**、**container-server**、**account-server**。

18.3. 文件权限

/var/lib/config-data/puppet-generated/swift/etc/swift/ 目录包含有关环拓扑和环境配置的信息。建议有以下权限：

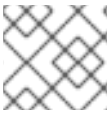
```
chown -R root:swift /var/lib/config-data/puppet-generated/swift/etc/swift/*
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type f -exec chmod 640 {} \;
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type d -exec chmod 750 {} \;
```

此限制仅允许 root 修改配置文件，同时仍然允许服务读取这些文件，因为其在 **swift** 组中成员资格。

18.4. 保护存储服务

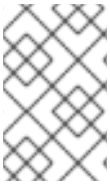
以下是各种存储服务的默认侦听端口：

- 帐户服务 - **TCP/6002**
- 容器服务 - **TCP/6001**
- Object Service - **TCP/6000**
- Rsync - **TCP/873**



注意

如果使用 ssync 而不是 rsync，则使用对象服务端口来维护其持久性。



注意

在存储节点中不会发生身份验证。如果您能够连接到其中一个端口中的存储节点，您可以在不进行身份验证的情况下访问或修改数据。为了帮助缓解这个问题，您应该遵循之前使用私有存储网络给出的建议。

18.5. OBJECT STORAGE ACCOUNT 术语

swift 帐户不是用户帐户或凭证。存在以下区别：

- Swift 帐户 - 容器的集合（非用户帐户或身份验证）。您使用的身份验证系统将决定哪些用户与帐户相关联，以及它们如何访问它。
- Swift 容器 - 对象集合。容器的元数据可用于 ACL。ACL 的使用取决于所使用的身份验证系统。
- Swift 对象 - 实际数据对象。对象级别的 ACL 也可用于元数据，并且依赖于所使用的身份验证系统。

在每个级别上，您都有控制用户访问的 ACL； ACL 根据正在使用的身份验证系统进行解释。最常见的身份验证提供程序类型是 Identity Service (keystone)；自定义身份验证提供程序也可用。

18.6. 保护代理服务

代理节点应至少有两个接口（物理或虚拟）：一个公共接口和一个私有。您可以使用防火墙或服务绑定来帮助保护公共接口。面向公共的服务是一个 HTTP Web 服务器，它处理端点客户端请求、验证它们并执行适当的操作。私有接口不需要任何侦听服务，而是用来建立与私有存储网络上存储节点的传出连接。

18.7. HTTP 侦听端口

director 将 Web 服务配置为在非 root 用户（无 UID 0）下运行。使用大于 **1024** 的端口号有助于避免以 root 身份运行 Web 容器的任何部分。通常，使用 HTTP REST API （并执行自动身份验证）的客户端将从身份验证响应中检索它们所需的完整 REST API URL。OpenStack REST API 允许客户端向一个 URL 进行身份验证，然后重新重定向为实际服务使用完全不同的 URL。例如：客户端可以对 **https://identity.cloud.example.org:55443/v1/auth** 进行身份验证，并使用其身份验证密钥和存储 URL（代理节点的 URL 或负载均衡器）获得响应 **https://swift.cloud.example.org:44443/v1/AUTH_8980。**

18.8. 负载均衡器

如果选项使用 Apache 不可行，或者您想要卸载 TLS 工作的性能，您可以使用专用的网络设备负载均衡器。这是使用多个代理节点时提供冗余和负载平衡的常见方法。

如果您选择卸载 TLS，请确保负载均衡器和代理节点之间的网络链接位于私有(V) LAN 网段上，以便网络上的其他节点（可能被入侵）不能对未加密的流量进行线tap （嗅探）。如果发生此类漏洞，攻击者可以访问端点客户端或云管理员凭证并访问云数据。

您使用的身份验证服务将决定如何在对端点客户端的响应中配置不同的 URL，允许它们使用负载均衡器而不是单独的代理节点。

18.9. 对象存储身份验证

Object Storage (swift)使用 WSGI 模型为仅提供一般可扩展性的中间件功能提供，也用于对端点客户端进行身份验证。身份验证供应商定义存在哪些角色和用户类型。有些使用传统用户名和密码凭证，另一些可能会利用 API 密钥令牌，甚至利用客户端 X.509 证书。可以使用自定义中间件集成自定义提供程序。

默认情况下，对象存储附带两个身份验证中间件模块，其中一个模块可用作开发自定义身份验证中间件的示例代码。

18.10. 加密 AT-REST SWIFT 对象

Swift 可以与 Barbican 集成，以透明地加密和解密您的存储(at-rest)对象。at-rest 加密与 in-transit 加密不同，引用在存储在磁盘上的对象。

Swift 以透明的方式执行这些加密任务，对象在上传到 swift 时会自动加密，然后在提供给用户时自动解密。这个加密和解密是使用相同的(symmetric)密钥完成的，该密钥存储在 Barbican 中。

其他资源

- 使用 OpenStack Key Manager 管理 secret

18.11. 其他项目

在每个节点上的 **/var/lib/config-data/puppet-generated/swift/etc/swift/swift.conf** 中，有一个 **swift_hash_path_prefix** 设置和一个 **swift_hash_path_suffix** 设置。它们是为了降低所存储对象的哈希冲突的可能性，并且验证一个用户覆盖其他用户的数据。

这个值最初应该使用加密的安全随机数字生成器设置，并在所有节点上保持一致。确保使用正确的 ACL 进行保护，并且您有备份副本以避免数据丢失。

第 19 章 监控和日志记录

日志管理是监控 OpenStack 部署安全状态的重要组件。除了组成 OpenStack 部署的组件活动外，日志还深入了解管理员、项目和实例的 BAU 操作。

日志不仅对主动安全性和持续合规活动有价值，它们也是调查和事件响应的宝贵信息源。例如，分析 keystone 访问日志可能会提醒您失败的登录、其频率、原始 IP 以及事件是否仅限于选择帐户，以及其他相关信息。

director 包括使用 AIDE 的入侵检测功能，以及 keystone 的 CADF 审计。如需更多信息，请参阅 强化基础架构和虚拟化。

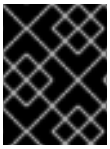
19.1. 强化监控基础架构

集中式日志记录系统是入侵者的高值目标，因为成功的破坏可能会允许它们清除或篡改事件记录。建议您使用以下命令强化监控平台。另外，请考虑对这些系统进行常规备份，并在出现中断或 DoS 时进行故障转移规划。

19.2. 要监控的事件示例

事件监控是一种更加主动的方法，用于保护环境，提供实时检测和响应。存在多个工具，它们有助于监控。对于 OpenStack 部署，您需要监控硬件、OpenStack 服务和云资源使用情况。

本节描述了您可能需要了解的一些示例事件。



重要

此列表并不完整。您需要考虑其他可能适用于您的特定网络的用例，并且您可能考虑异常行为。

- 检测没有生成日志的事件是高值的事件。这类差距可能表示服务故障，甚至是临时关闭日志记录或修改日志级别来隐藏其跟踪的入侵者。
- 应用程序事件（如启动或停止事件）可能会造成安全隐患。
- OpenStack 节点上的操作系统事件，如用户登录或重启。它们可以提供宝贵的见解，以区分正常和不当的系统使用。
- 网络桥接关闭。由于服务中断的风险，这是一个可操作的事件。
- iptables 清除 Compute 节点上的事件，以及导致对实例的访问丢失。

为了降低在身份服务中用户、项目或删除孤立实例的安全风险，需要讨论系统中生成通知，并使 OpenStack 组件根据情况响应这些事件，如终止实例、断开连接的卷、回收 CPU 和存储资源等。

安全监控控制，如入侵检测软件、防病毒软件和删除实用程序，可以生成日志来显示攻击或入侵的发生情况。这些工具可以在 OpenStack 节点上部署时提供一层保护。项目用户可能还想在其实例上运行此类工具。

第 20 章 项目的数据隐私

OpenStack 旨在支持具有不同数据要求的项目之间的多租户。云操作员需要考虑其适用的数据隐私问题和法规。本章解决了 OpenStack 部署的数据所在的方面。

20.1. 数据驻留

在过去几年中，数据的隐私和隔离一直被认为是云采用的主要障碍。对谁拥有云中的数据以及云操作员最终是否能够信任作为这些数据的 custodian 在过去存在重大问题。

某些 OpenStack 服务可以访问属于项目的数据和元数据，或者引用项目信息。例如，存储在 OpenStack 云中的项目数据可能包括以下项目：

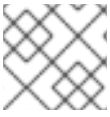
- 对象存储对象。
- 计算实例临时文件系统存储.
- 计算实例内存.
- 块存储卷数据。
- 用于计算访问的公钥。
- 镜像服务中的虚拟机镜像。
- 实例快照.
- 传递给计算配置驱动器扩展的数据。

OpenStack 云存储的元数据包括以下项目（此列表不准确）：

- 机构名称。
- 用户的"真实名称"。
- 运行的实例、存储桶、对象、卷和其他与配额相关的项目的数量或大小。
- 运行的实例或存储数据的小时数。
- 用户的 IP 地址。
- 内部为计算镜像绑定生成的私钥。

20.2. 数据处理

良好的实践建议，Operator 在处理机构控制前必须清理云系统介质（数字和非数字），然后再发布机构控制，或在发布前重复使用。由于信息的特定安全域和敏感度，应实施适当的强度和完整性级别。



注意

NIST Special Publication 800-53 Revision 4 在本主题上获取特定的视图：

> 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.

云操作员在开发通用数据分布和清理指南（根据 NIST 推荐的安全控制）时应考虑以下事项：

- 跟踪、记录和验证媒体清理和配件操作。
- 测试清理设备和程序以验证性能是否正确。
- 在将此类设备连接到云基础架构之前，清理可移植、可移动存储设备。
- 销毁无法清理的云系统介质。

因此，OpenStack 部署需要解决以下实践（另外一项）：

- 安全数据纠删代码
- 实例内存清理
- 块存储卷数据
- 计算实例临时存储
- 裸机服务器清理

20.2.1. 没有被安全清除的数据

在 OpenStack 中，可能会删除一些数据，但无法象上述 NIST 标准上下文中那样安全地清除。这通常适用于存储在数据库中的最大或全部定义元数据和信息。这可以通过用于自动 vacuuming 和 periodic free-space wiping 的数据库和/或系统配置修复。

20.2.2. 实例内存清理

特定于各种虚拟机监控程序是实例内存的处理。在 Compute 中不定义此行为，虽然虚拟机监控程序通常预计在删除实例时、实例或两者时清理内存是尽力。

20.3. 加密 CINDER 卷数据

强烈建议使用 OpenStack 卷加密功能。在 Volume Encryption 下的 Data Encryption 部分中将对此进行讨论。使用此功能时，通过安全地删除加密密钥来实现数据破坏。最终用户可以在创建卷时选择此功能，但请注意，管理员必须首先执行一次性设置卷加密功能。

如果没有使用 OpenStack 卷加密功能，则其他方法通常更难以启用。如果使用后端插件，则可能独立进行加密或非标准覆盖解决方案。OpenStack 块存储的插件以各种方式存储数据。许多插件都特定于某个供应商或技术，而其他插件则特定于文件系统的 DIY 解决方案（如 LVM 或 ZFS）。安全销毁数据的方法因插件、供应商和文件系统而异。

有些后端（如 ZFS）支持写时复制，以防止数据暴露。在这些情况下，从未写入块读取始终为零。其他后端（如 LVM）可能无法原生支持，因此 cinder 插件在向用户移交之前覆盖之前写入的块。务必要检查您选择的卷后端提供什么保证，以及查看未提供这些保证的补救可能可用。

20.4. 镜像服务延迟删除功能

镜像服务具有延迟删除功能，该功能将在定义的时间段内删除镜像。如果此行为是安全问题，请考虑禁用此功能；您可以通过编辑 **glance-api.conf** 文件并将 **delayed_delete** 选项设置为 **False** 来执行此操作。

20.5. 计算软删除功能

compute 具有软删除功能，它允许在定义的时间段内处于软删除状态的实例。该实例可在此时间段内恢复。要禁用 soft-delete 功能，请编辑 **/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf** 文件，并将 **reclaim_instance_interval** 选项留空。

20.6. 裸机置备的安全强化

对于裸机置备基础架构，您应该考虑安全强化基板管理控制器(BMC)，特别是 IPMI。例如，您可以在置备网络中隔离这些系统，配置非默认和强大的密码，并禁用不需要的管理功能。如需更多信息，请参阅供应商有关安全强化这些组件的指导。

注意

如果可能，请考虑评估基于 Redfish 的 BMC，而不是旧的 BMC。

20.7. 硬件识别

在部署服务器时，可能并不总是有可靠的方法将其与攻击者的服务器区分开来。这个功能可能依赖于硬件/BMC 到某种程度，但通常看似没有可验证的方法在服务器中内置。

20.8. 数据加密

使用 选项可以加密在磁盘上存储或通过网络传输的项目数据，如下面描述的 OpenStack 卷加密功能。这高于常规建议，用户在将其发送到供应商前对自己的数据进行加密。

代表项目加密数据的重要性与攻击者可以访问项目数据的供应商承担的风险密切相关。政府可能有一些要求，以及按政策、私人合同等要求，甚至在遵守公有云提供商的私人合同时也是如此。在选择项目加密策略前，请考虑获得风险评估和法律建议。

per-instance 或 per-object 加密会首选使用，按按项目、每个项目和每个云聚合降序排列。本建议涉及实施的复杂性和难度。目前，对于某些项目，实施加密非常困难或根本不可能。实施者应考虑加密项目数据。

通常，数据加密与通过丢弃密钥而可靠地销毁项目和每个实例数据的功能有积极的影响。请注意，在这样做时，以可靠和安全的方式销毁这些密钥会变得非常重要。

向用户加密数据的机会：

- Object Storage 对象
- 网络数据

20.8.1. 卷加密

OpenStack 中的卷加密功能支持基于每个项目的隐私。支持以下功能：

- 通过控制面板或命令行界面启动的加密卷类型创建和使用
- 启用加密并选择参数，如加密算法和密钥大小
- iSCSI 数据包中包含的卷数据是加密的
- 如果原始卷加密，支持加密备份

- 仪表板显示卷加密状态。包括表示卷已加密，并包含加密参数，如算法和密钥大小
- 使用密钥管理服务接口

20.8.2. Object Storage 对象

Object Storage (swift)支持存储节点上静态对象数据的可选加密。对象数据的加密旨在降低在未授权方获得对磁盘的物理访问时读取用户数据的风险。

对静态数据进行加密是由中间件实现的，这些中间件可能包含在代理服务器 WSGI 管道中。这个功能是 swift 集群内部，无法通过 API 公开。客户端不知道，此功能在内部向 swift 服务加密数据；内部加密数据不应通过 swift API 返回到客户端。

以下数据在 swift 中被加密。

- 对象内容，例如，对象 **PUT** 请求正文的内容。
- 具有非零内容的对象的实体标签(**ETag**)。
- 所有自定义用户对象元数据值。例如，使用带有 **X-Object-Meta-** 前缀的标头的 **PUT** 或 **POST** 请求发送元数据。

以上列表中不包含的任何数据或元数据都不会加密，包括：

- 帐户、容器和对象名称
- 帐户和容器自定义用户元数据值
- 所有自定义用户元数据名称
- 对象 Content-Type 值
- 对象大小
- 系统元数据

20.8.3. 块存储性能和后端

在启用操作系统时，您可以使用 Intel 和 AMD 处理器中提供的硬件加速功能来提高 OpenStack 卷加密性能。

OpenStack 卷加密功能使用主机上的 **dm-crypt** 或原生 **QEMU** 加密支持来保护卷数据。红帽建议您在创建加密卷时使用 **LUKS** 卷加密类型。

20.8.4. 网络数据

Compute 节点的项目数据可以通过 IPsec 或其他隧道进行加密。这种做法在 OpenStack 中不是常见或标准，但可以选择动机和感兴趣的实施人员。同样，加密数据仍然加密，因为它通过网络传输。

20.9. 密钥管理

为了解决项目数据隐私的经常提到的问题，OpenStack 社区对数据进行加密有很大兴趣。在将数据保存到云之前，最终用户会相对容易加密其数据，而这是项目对象（如媒体文件、数据库存档等）的可行路径。在某些情况下，客户端加密用于加密需要客户端交互（如显示密钥）来解密数据以供以后使用的数据。

barbican 可帮助项目更无缝地加密数据，并使其可访问，而不给用户造成密钥管理。作为 OpenStack 的一个部分提供加密和密钥管理服务，可简化数据采用，并有助于解决客户对隐私或数据滥用的问题。

卷加密功能依赖于密钥管理服务，如密钥管理器服务(barbican)，用于创建和安全强化的密钥存储。

第 21 章 管理实例安全性

在虚拟化环境中运行实例的一个好处是，安全控制在部署到裸机时通常不提供的新机会。某些技术可应用于虚拟化堆栈，从而提高了 OpenStack 部署的信息保障。具有强大安全要求的 Operator 可能需要考虑部署这些技术，但并非所有 Operator 都适用。在某些情况下，由于存在性业务要求，在云中使用技术可能已被排除。同样，一些技术会检查实例数据，如运行状态，这可能不适合系统用户。

本章描述了这些技术以及可用于帮助提高实例或底层节点安全性的情况。可能的隐私顾虑也会突出显示，其中包括数据透传、内省或熵源。

21.1. 为实例提供熵

熵指的是可供实例使用的随机数据的质量和源。加密技术通常依赖于随机性，这需要从熵池中提取。当实例无法有足够的熵来支持加密技术所需的随机性时，会进行熵。熵可在实例中作为看似不相关的内容进行清单。例如，等待 SSH 密钥生成的实例可能会引起较慢的引导时间。熵不足的可能性还会导致云用户使用来自实例内部的质量熵源，从而使在云中运行的应用程序更加安全。

为了向实例提供高质量的熵来源，您需要在云中有足够的硬件随机数生成器(HRNG)来支持实例。对于日常操作，现代 HRNG 可以生成足够的熵来支持 50 到 100 个 Compute 节点。高带宽 HRNG 可以处理更多节点。您必须识别云的应用程序要求，以确保有足够的熵可用。

VirtIO RNG 是一个随机数生成器，它默认使用 **/dev/urandom** 作为熵的来源，以确保实例在引导时不会耗尽熵。也可以将其配置为使用 HRNG 或像熵收集守护进程(EGD)等工具来通过部署分发熵。virtio RNG 设备默认为实例启用。要为实例禁用 Virtio RNG 设备，您必须在实例类别上将 **hw_rng:allowed** 设置为 **False**。

21.2. 将实例调度到节点

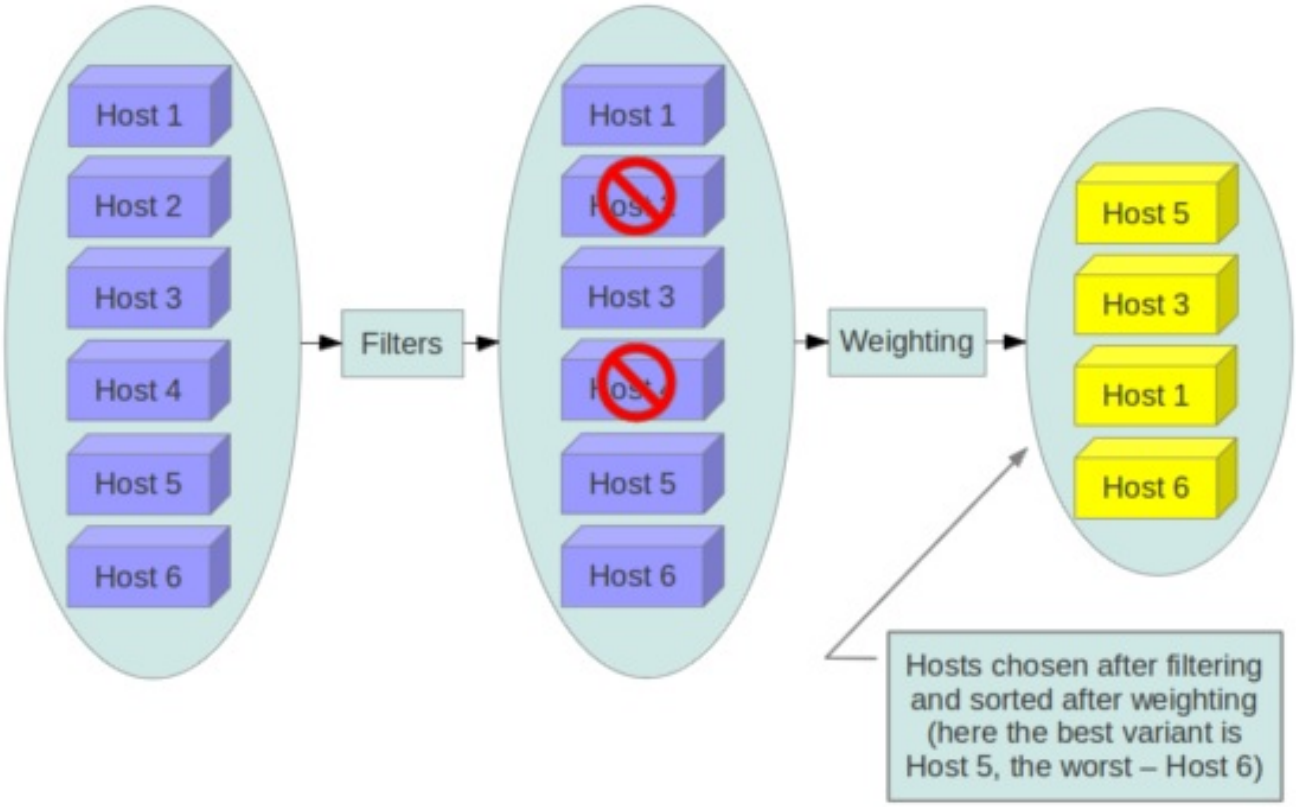
在创建实例前，必须选择镜像实例化的主机。此选择由 **nova-scheduler** 执行，它决定了如何分配计算和卷请求。

FilterScheduler 是计算的默认调度程序，但存在其他调度程序。此功能与 *过滤器提示* 配合使用，以确定实例应启动的位置。通过这一主机选择，管理员可以满足许多不同的安全性和合规性要求。如果数据隔离是主要关注，您可以选择让项目实例尽可能驻留在同一主机上。相反，您可以因为可用性或容错原因，尝试让实例位于尽可能多的不同主机上。

过滤调度程序属于以下主要类别：

- *基于资源的过滤器* - 根据 hypervisor 主机集的系统资源使用情况来确定实例的放置，并可以在空闲或使用的属性上触发，如 RAM、IO 或 CPU 使用率。
- *基于镜像的过滤器* - 根据所使用的镜像元数据（如虚拟机或镜像类型的操作系统）创建实例。
- *基于环境的过滤器* - 根据外部详细信息（如特定 IP 范围内、跨可用性区域或与另一个实例在同一主机上）来确定实例的放置。
- *自定义条件* - 根据用户或管理员提供的标准（如信任或元数据解析）来限制实例创建。

可以一次应用多个过滤器。例如，**ServerGroupAffinity** 过滤器检查是否在特定主机的成员上创建实例，并且 **ServerGroupAntiAffinity** 过滤器检查是否在另一特定的一组主机上创建相同的实例。请注意，这两个过滤器通常同时启用，且永远不会相互冲突，因为它们都检查给定属性值，且不能同时为 true。



重要

考虑禁用可解析用户提供的对象的过滤器，或者可以操作（如元数据）。

21.3. 使用可信镜像

在云环境中，用户使用预安装的镜像或其上传本身的镜像。在这两种情况下，用户应该能确保他们所使用的镜像未被篡改。验证镜像的功能是安全性的根本性。需要信任链，从镜像源到使用它的目标。这可以通过签名从可信源获取的镜像，并在使用前验证签名来完成。下面将讨论获取和创建验证镜像的各种方法，后跟镜像签名验证功能的描述。

21.4. 创建镜像

有关如何创建和管理镜像到 Red Hat OpenStack Image 服务(glance)的指导，请参阅创建和管理镜像。为您的环境使用可信镜像以提高安全性，并使用您机构的强化指南来进一步保护。您可以通过以下几种方法之一获取您的环境的镜像：

下载实例介质

要从可信源获取引导介质，请从官方红帽源下载镜像，并使用 SHA256SUM 来验证。

从 ISO 创建镜像

有关从安装过程创建镜像的详情，请参考 创建 Red Hat Enterprise Linux 9 镜像 。

使用镜像构建器

您可以使用 **disk-image-builder** 生成仅出于 OpenStack 中目的需要组件的最小系统。有关使用 **disk-image-builder** 创建自定义镜像的详情，请参考 生成自定义的 RHEL 系统镜像。

21.5. 验证镜像签名

您可以启用镜像签名验证，以确保在 Compute 服务(nova)启动实例前，镜像服务(glance)镜像不包含未经授权的更改。启用这个功能后，您可以防止新实例启动，其中可能包括恶意软件或安全漏洞。

先决条件

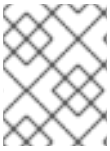
- 已安装 Red Hat OpenStack Platform director 环境。
- 以 stack 身份登录 director。

流程

1. 在 heat 模板中，通过将 **True** 的值设置为 **VerifyGlanceSignatures** 参数来启用实例签名验证：

```
parameter_defaults:
  VerifyGlanceSignatures: True
```

2. 确保用于修改 **VerifyGlanceSignatures** 参数的模板包含在 **openstack overcloud deploy** 脚本中，然后重新运行部署脚本。



注意

如果您使用没有签名的镜像创建实例，则镜像会失败，且实例不会启动。有关签名镜像的更多信息，请参阅签名镜像。

21.6. 迁移实例

OpenStack 和底层虚拟化层为在 OpenStack 节点之间实时迁移镜像，允许您无缝地执行 Compute 节点的滚动升级，而无需实例停机。但是，实时迁移也存在很大的风险。要了解涉及的风险，以下是实时迁移过程中执行的高级别步骤：

1. 在目标主机上启动实例
2. 传输内存
3. 停止客户机和同步磁盘
4. 传输状态
5. 启动客户机

注意

某些操作（如冷迁移、调整大小和 shelve）都可能导致实例的数据传送到其他服务等。

21.6.1. 实时迁移风险

在实时迁移过程的不同阶段，实例运行时间内存和磁盘的内容以纯文本形式通过网络传输。因此，使用实时迁移时需要解决多个风险。以下非详细列表详细介绍了其中的一些风险：

- 拒绝服务(DoS)：如果在迁移过程中出现某种失败，则实例可能会丢失。
- 数据暴露：必须安全处理内存或磁盘加密。

- 数据操作：如果内存或磁盘传输没有被安全处理，则攻击者可以在迁移过程中操作用户数据。
- 代码注入：如果内存或磁盘传输没有被安全处理，则攻击者可以在迁移过程中处理可执行文件，可以是磁盘或内存中的可执行文件。

21.6.2. 禁用实时迁移

目前，OpenStack 中默认启用实时迁移。默认情况下，实时迁移是仅 admin 的任务，因此用户无法启动此操作，只有管理员（假定为可信）。通过在 nova **policy.json** 文件中添加以下行，可以禁用实时迁移：

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

或者，当阻止 TCP 端口 **49152** 到 **49261** 时，实时迁移应该会失败，或者确保 nova 用户在计算主机之间没有免密码 SSH 访问。

请注意，实时迁移的 SSH 配置被锁定：新用户已创建(nova_migration)，SSH 密钥仅限于该用户，并且仅在允许的网络中使用。然后，打包程序脚本会限制可以运行的命令（例如，在 libvirt 套接字上的 netcat）。

21.6.3. 加密实时迁移

实时迁移流量以纯文本形式传输正在运行的实例的磁盘和内存内容，且当前默认托管在内部 API 网络中。

如果有足够的要求（如升级）来保持启用实时迁移，则 libvirtd 可以为实时迁移提供加密的隧道。但是，此功能不会在 OpenStack Dashboard 或 nova-client 命令中公开，只能通过 libvirtd 的手动配置访问。然后，实时迁移过程会改为以下高级别步骤：

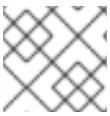
1. 实例数据从 hypervisor 复制到 libvirtd。
2. 加密隧道在源和目标主机上的 libvirtd 进程之间创建。
3. 目标 libvirtd 主机将实例复制到底层虚拟机监控程序。

注意

对于 Red Hat OpenStack Platform 13，推荐的方法是使用隧道迁移，该迁移在将 Ceph 用作后端时默认启用。如需更多信息，请参阅 https://docs.openstack.org/nova/queens/configuration/config.html#libvirt.live_migration_t

21.7. 监控、警报和报告

实例是可在主机之间复制的服务器镜像。因此，最好在物理和虚拟主机之间应用日志。应记录操作系统和应用程序事件，包括访问主机和数据的事件、用户添加和删除、特权更改等，具体由您的要求指定。考虑将结果导出到收集日志事件的日志聚合器，关联它们进行分析，并存储它们以引用或进一步操作。一个常见的工具是 ELK 堆栈或 Elasticsearch、Logstash 和 Kibana。



注意

这些日志应定期检查，甚至可以在网络操作中心(NOC)执行的实时视图中监控。

您需要进一步确定哪些事件将触发随后发送到操作响应器的警报。

21.8. 更新和补丁

管理程序运行独立的虚拟机。这个虚拟机监控程序可以在操作系统中运行，也可以直接在硬件上运行（称为裸机）。对虚拟机监控程序的更新不会传播到虚拟机。例如，如果部署正在使用 KVM 并具有一组 CentOS 虚拟机，则对 KVM 的更新不会更新 CentOS 虚拟机上运行的任何内容。

考虑将明确的虚拟机的所有权分配给所有者，然后负责虚拟机的强化、部署和持续功能。您还应该有一个定期部署更新的计划，同时首先在类似生产环境的环境中测试更新。

21.9. 防火墙和实例配置集

最常见的操作系统包括基于主机的防火墙用于额外的安全层。虽然实例应尽可能多地运行（在单一用途实例的点上），在实例上运行的所有应用程序都应进行配置，以确定应用需要访问的系统资源、运行所需的最低特权级别，以及预期的网络流量。此预期流量应当作为允许的流量添加到基于主机的防火墙中，以及 SSH 或 RDP 等任何必要的日志记录和管理通信。所有其他流量都应在防火墙配置中明确拒绝。

在 Linux 实例上，上述应用程序配置文件可与 **audit2allow** 等工具一起使用，以构建 SELinux 策略，进一步保护大多数 Linux 发行版上的敏感系统信息。SELinux 使用用户、策略和安全上下文的组合来划分应用运行所需的资源，并将其与不需要的其他系统资源进行分段。

注意

Red Hat OpenStack Platform 默认启用了 SELinux，策略是为 OpenStack 服务自定义的。根据需要，请考虑定期检查这些策略。

21.10. 安全组

OpenStack 为主机和网络提供安全组，以深入了解给定项目中的实例。它们与基于主机的防火墙类似，因为它们根据端口、协议和地址允许或拒绝传入流量。但是，安全组规则仅应用于传入流量，而基于主机的防火墙规则则可应用于传入和传出流量。主机和网络安全组规则也可以冲突和拒绝合法流量。考虑检查安全组是否针对所使用的网络进行了正确配置。如需了解更多详细信息，请参阅本指南中的安全组。

注意

除非特别需要禁用安全组和端口安全性，否则您应保持安全组和端口安全性。要根据明确的深度方法构建，建议您将细粒度规则应用到实例。

21.11. 访问实例控制台

默认情况下，实例的控制台可通过虚拟控制台远程访问。这可用于故障排除目的。Red Hat OpenStack Platform 使用 VNC 进行远程控制台访问。

- 考虑使用防火墙规则锁定 VNC 端口。默认情况下，**nova_vnc_proxy** 使用 **6080** 和 **13080**。
- 确认 VNC 流量由 TLS 加密。对于基于 director 的部署，以 **UseTLSTransportForVnc** 开始。

21.12. 证书注入

如果需要 SSH 到实例，您可以将 Compute 配置为在创建时自动将所需的 SSH 密钥注入到实例中。

其他资源

- 将 KVM 客户机镜像与 Red Hat OpenStack Platform 搭配使用 。

第 22 章 消息队列

消息队列服务有助于在 OpenStack 中进行进程间通信。这使用这些消息排队服务后端完成：

- RabbitMQ - Red Hat OpenStack Platform 默认使用 RabbitMQ。
- qpid

RabbitMQ 和 Qpid 都是高级消息队列协议(AMQP)框架，它为对等通信提供消息队列。队列实施通常部署为集中或分散的队列服务器池。

消息队列有效地促进 OpenStack 部署之间的命令和控制功能。允许访问队列后，不会执行进一步的授权检查。通过队列访问的服务会验证实际消息有效负载内的上下文和令牌。但是，您必须注意令牌的过期日期，因为令牌可能会重新显示，并可授权基础架构中的其他服务。

OpenStack 不支持消息级信任，如消息签名。因此，您必须保护并验证消息传输本身。对于高可用性(HA)配置，您必须执行队列到队列身份验证和加密。

22.1. 消息传递传输安全性

基于 AMQP 的解决方案(Qpid 和 RabbitMQ)支持使用 TLS 的传输级安全性。

考虑为您的消息队列启用传输级别加密。使用 TLS 进行消息传递客户端连接，可以保护通信免受篡改，并窃取传输至消息传递服务器的通信。以下指导通常如何为两个流行消息传递服务器配置 TLS： Qpid 和 RabbitMQ。当您配置消息传递服务器用来验证客户端连接的可信证书颁发机构(CA)捆绑包时，建议只限于用于节点的 CA，最好是内部管理的 CA。可信 CA 的捆绑包将决定哪些客户端证书被授权，并传递设置 TLS 连接的 client-server 验证步骤。

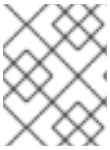
注意

安装证书和密钥文件时，请确保对文件权限进行限制，例如使用 **chmod 0600**，并且所有权仅限于消息传递服务器守护进程用户，以防止由其他进程和消息传递服务器上的用户进行未授权访问。

22.1.1. RabbitMQ 服务器 SSL 配置

以下行应添加到系统范围的 RabbitMQ 配置文件中，通常为 **/etc/rabbitmq/rabbitmq.config** ：

```
[
  {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}]}
   ]}
].
```



注意

tcp_listeners 选项被设置为 **[]**，以防止它侦听非 SSL 端口。**ssl_listeners** 选项应限制为仅侦听服务的管理网络。

22.2. 队列身份验证和访问控制

RabbitMQ 和 Qpid 提供控制对队列的访问的身份验证和访问控制机制。

简单身份验证和安全层(SASL)是用于在互联网协议中进行身份验证和数据安全的框架。RabbitMQ 和 Qpid 均提供 SASL 和其他可插拔身份验证机制，除了允许增强身份验证安全性的简单用户名和密码之外。虽然 RabbitMQ 支持 SASL，但 OpenStack 中的支持目前不允许请求特定的 SASL 身份验证机制。OpenStack 中的 RabbitMQ 支持允许通过未加密的连接或用户名和密码以及 X.509 客户端证书进行身份验证，以建立安全的 TLS 连接。

考虑在所有 OpenStack 服务节点上为客户端连接消息传递队列配置 X.509 客户端证书，并在可能的情况下使用 X.509 客户端证书执行身份验证。使用用户名和密码时，应为每个服务创建帐户，以及用于精细访问队列的细粒度可审核性。

在部署之前，请考虑排队服务器使用的 TLS 库。Qpid 使用 Mozilla 的 NSS 库，而 RabbitMQ 使用 Erlang 的 TLS 模块来使用 OpenSSL。

22.3. RABBITMQ 的 OPENSTACK 服务配置

本节介绍 OpenStack 服务的典型 RabbitMQ 配置：

```
[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
```

注意

将 **RABBIT_PASS** 替换为合适的密码。

22.4. QPID 的 OPENSTACK 服务配置

本节介绍 OpenStack 服务的典型 Qpid 配置：

```
[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
```

注意

使用合适的密码替换 **QPID_PASS**。

另外，如果将 SASL 与 Qpid 一起使用，则可以通过添加以下内容指定要使用的 SASL 机制：

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

22.5. 消息队列进程隔离和策略

每个项目提供许多服务来发送和接收消息。每个发送消息的二进制文件都应该消耗来自队列的消息（如果只有回复）。

消息队列服务进程应相互隔离，以及机器上的其他进程。

22.6. 命名空间

Linux 使用命名空间将进程分配到独立域。本指南的其他部分将更详细地涵盖系统划分。

第 23 章 保护 RED HAT OPENSTACK PLATFORM 中的端点

与 OpenStack 云互动的过程从查询 API 端点开始。虽然公共和私有端点存在不同的挑战，但这些价值资产在被破坏时可能会带来很大的风险。

本章推荐对公共和面向私有的 API 端点进行安全增强。

23.1. 内部 API 通信

OpenStack 提供面向公共的内部管理和私有 API 端点。默认情况下，OpenStack 组件使用公开定义的端点。建议将这些组件配置为使用正确的安全域中的 API 端点。内部管理端点允许进一步提升对 keystone 的访问，因此可能需要进一步隔离它。

服务根据 OpenStack 服务目录选择其对应的 API 端点。这些服务可能没有遵循列出的公共或内部 API 端点值。这可能导致内部管理流量路由到外部 API 端点。

23.2. 在 IDENTITY 服务目录中配置内部 URL

Identity 服务目录应该了解您的内部 URL。虽然默认情况下不使用此功能，但可以通过配置来提供该功能。另外，当此行为变为默认值后，它应该与预期更改进行向前兼容。

考虑将配置的端点与网络级别隔离，只要它们具有不同的访问权限级别。Admin 端点供云管理员访问，因为它提供对内部或公共端点上不可用的 keystone 操作的提升访问权限。内部端点设计为使用云的内部（例如，由 OpenStack 服务），通常无法在部署网络外访问。公共端点应该启用 TLS，且部署外的唯一 API 端点可以访问供云用户操作。

director 自动注册端点的内部 URL。如需更多信息，请参阅 https://github.com/openstack/tripleo-heat-templates/blob/a7857d6dfcc875eb2bc611dd9334104c18fe8ac6/network/endpoints/build_endpoint_map

23.3. 为内部 URL 配置应用程序

您可以强制某些服务使用特定的 API 端点。因此，建议任何联系另一个服务的 API 的 OpenStack 服务都必须被显式配置，以访问正确的内部 API 端点。

每个项目可能会以不一致的方式定义目标 API 端点。OpenStack 的未来发行版本希望通过统一地使用身份服务目录来解决这些不一致的问题。

23.4. 粘贴和中间件

OpenStack 中的大多数 API 端点和其他 HTTP 服务都使用 Python Paste Deploy 库。从安全的角度来看，这个库允许通过应用程序的配置对请求过滤器管道进行操作。此链中的每一元素称为中间件。在管道中更改过滤器顺序或添加额外的中间件可能会对安全影响不可预测的安全影响。

通常，实施者会添加中间件来扩展 OpenStack 的基本功能。考虑仔细考虑在 HTTP 请求管道中添加非标准软件组件所引入的潜在风险。

23.5. 安全 METADEF API

在 Red Hat OpenStack Platform (RHOSP)中，云管理员可以使用元数据定义(metadef) API 定义键值对和标签元数据。对云管理员可以创建的 metadef 命名空间、对象、属性、资源或标签的数量没有限制。

镜像服务策略控制 metadef API。默认情况下，只有云管理员才能创建、更新或删除(CUD) metadef API。这个限制可防止 metadef API 向未授权用户公开信息，并减少恶意用户用无限资源填充镜像服务(glance)数据库的风险，从而可以创建拒绝服务(DoS)的攻击。但是，云管理员可覆盖默认策略。

23.6. 为云用户启用 METADEF API 访问

具有依赖于元数据定义(metadef) API 写入访问权限的云管理员可以通过覆盖默认的 admin-only 策略来使所有用户访问这些 API。但是，在这种配置中，可能会意外泄漏敏感资源名称，如客户名称和内部项目。管理员必须审核其系统，以识别之前创建的资源，这些资源可能容易受到攻击，即使所有用户只启用了读写。

流程

1. 作为云管理员，登录 undercloud 并为策略覆盖创建一个文件。例如：

```
$ cat open-up-glance-api-metadef.yaml
```

2. 配置策略覆盖文件，以允许对所有用户进行 metadef API 读写访问：

```
GlanceApiPolicies: {
    glance-metadef_default: { key: 'metadef_default', value: '' },
    glance-get_metadef_namespace: { key: 'get_metadef_namespace', value:
'rule:metadef_default' },
    glance-get_metadef_namespaces: { key: 'get_metadef_namespaces', value:
'rule:metadef_default' },
    glance-modify_metadef_namespace: { key: 'modify_metadef_namespace', value:
'rule:metadef_default' },
    glance-add_metadef_namespace: { key: 'add_metadef_namespace', value:
'rule:metadef_default' },
    glance-delete_metadef_namespace: { key: 'delete_metadef_namespace', value:
'rule:metadef_default' },
    glance-get_metadef_object: { key: 'get_metadef_object', value: 'rule:metadef_default' },
    glance-get_metadef_objects: { key: 'get_metadef_objects', value: 'rule:metadef_default' },
    glance-modify_metadef_object: { key: 'modify_metadef_object', value:
'rule:metadef_default' },
    glance-add_metadef_object: { key: 'add_metadef_object', value: 'rule:metadef_default' },
    glance-delete_metadef_object: { key: 'delete_metadef_object', value: 'rule:metadef_default'
},
    glance-list_metadef_resource_types: { key: 'list_metadef_resource_types', value:
'rule:metadef_default' },
    glance-get_metadef_resource_type: { key: 'get_metadef_resource_type', value:
'rule:metadef_default' },
    glance-add_metadef_resource_type_association: { key:
'add_metadef_resource_type_association', value: 'rule:metadef_default' },
    glance-remove_metadef_resource_type_association: { key:
'remove_metadef_resource_type_association', value: 'rule:metadef_default' },
    glance-get_metadef_property: { key: 'get_metadef_property', value: 'rule:metadef_default'
},
    glance-get_metadef_properties: { key: 'get_metadef_properties', value:
'rule:metadef_default' },
    glance-modify_metadef_property: { key: 'modify_metadef_property', value:
'rule:metadef_default' },
    glance-add_metadef_property: { key: 'add_metadef_property', value: 'rule:metadef_default'
},
    glance-remove_metadef_property: { key: 'remove_metadef_property', value:
```

```
'rule:metadef_default' },
    glance-get_metadef_tag: { key: 'get_metadef_tag', value: 'rule:metadef_default' },
    glance-get_metadef_tags: { key: 'get_metadef_tags', value: 'rule:metadef_default' },
    glance-modify_metadef_tag: { key: 'modify_metadef_tag', value: 'rule:metadef_default' },
    glance-add_metadef_tag: { key: 'add_metadef_tag', value: 'rule:metadef_default' },
    glance-add_metadef_tags: { key: 'add_metadef_tags', value: 'rule:metadef_default' },
    glance-delete_metadef_tag: { key: 'delete_metadef_tag', value: 'rule:metadef_default' },
    glance-delete_metadef_tags: { key: 'delete_metadef_tags', value: 'rule:metadef_default' }
  }
```

注意

您必须将所有 metadef 策略配置为使用 **rule:metadeta_default**。

3. 在部署 overcloud 时，请使用 **-e** 选项将新的策略文件包含在部署命令中：

```
$ openstack overcloud deploy -e open-up-glance-api-metadef.yaml
```

23.7. 更改 HAPROXY 的 SSL/TLS 密码和规则

如果您在 overcloud 中启用了 SSL/TLS，请考虑强化与 HAProxy 配置一起使用的 SSL/TLS 密码和规则。通过强化 SSL/TLS 密码，您可以避免 SSL/TLS 漏洞，如 POODLE 漏洞。

1. 创建名为 **tls-ciphers.yaml** 的 heat 模板环境文件：

```
touch ~/templates/tls-ciphers.yaml
```

2. 使用环境文件中的 **ExtraConfig** hook 将值应用到 **tripleo::haproxy::ssl_cipher_suite** 和 **tripleo::haproxy::ssl_options** hieradata:

```
parameter_defaults:
  ExtraConfig:
    tripleo::haproxy::ssl_cipher_suite:: `DHE-RSA-AES128-CCM:DHE-RSA-AES256-
CCM:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-
AES128-CCM:ECDHE-ECDSA-AES256-CCM:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305`

    tripleo::haproxy::ssl_options:: no-sslv3 no-tls-tickets
```

注意

cipher 集合是一个连续行。

3. 在部署 overcloud 时，使用 overcloud deploy 命令包含 **tls-ciphers.yaml** 环境文件：

```
openstack overcloud deploy --templates \
...
-e /home/stack/templates/tls-ciphers.yaml
...
```

23.8. 网络策略

API 端点通常跨越多个安全区，因此您必须特别注意 API 进程分离。例如，在网络设计级别，您可以考虑仅限制对指定系统的访问。如需更多信息，请参阅有关安全区的指导。

通过仔细建模，您可以使用网络 ACL 和 IDS 技术来强制实施网络服务之间的显式点对点通信。作为重要的跨域服务，这种明确执行适用于 OpenStack 的消息队列服务。

要强制执行策略，您可以配置服务、基于主机的防火墙（如 iptables）、本地策略(SELinux)以及可选的全局网络策略。

23.9. 强制访问控制

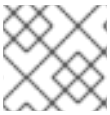
您应该将 API 端点进程与计算机上的其他进程隔离开来。这些进程的配置应该通过 Discretionary Access Controls (DAC)和强制访问控制(MAC)限制。这些增强的访问控制的目标是帮助包含 API 端点安全漏洞。

23.10. API 端点速率限制

速率限制是一种控制基于网络的应用接收的事件频率的方法。如果没有强大的速率限制，可能会导致应用程序容易受到各种拒绝服务攻击。对于 API，对于 API 来说尤其如此，其性质旨在接受类似请求类型和操作的高频率。

建议所有端点（特别是公共）都提供额外的保护层，例如，使用物理网络设计、速率限制代理或 Web 应用防火墙。

在配置和实施任何速率限制功能时，操作员仔细计划并考虑其 OpenStack 云内用户和服务所需的个人性能需求。



注意

对于 Red Hat OpenStack Platform 部署，所有服务都放置在负载平衡代理后。

第 24 章 实施 FEDERATION

红帽支持将 Red Hat Single Sign-on (RH-SSO)或 Microsoft Active Directory Federation Services (AD FS)与 Red Hat OpenStack Platform (RHOSP)结合使用。

24.1. 使用红帽单点登录与 IDM 联邦

您可以使用 Red Hat Single Sign-On (RH-SSO)来联合 IdM 用户进行 OpenStack 身份验证(authN)。联邦（联邦）可让您的 IdM 用户登录 OpenStack 控制面板，而无需向任何 OpenStack 服务显示其凭证。相反，当控制面板需要用户的凭证时，它会将用户转发到 RH-SSO，并允许他们在此处输入其 IdM 凭证。因此，RH-SSO 会返回到用户已成功验证的 Dashboard，控制面板则允许用户访问项目。

24.2. 联邦工作流

本节论述了 Identity 服务(keystone)、RH-SSO 和 IdM 相互交互的方式。OpenStack 中的联邦使用身份提供程序和服务提供商的概念：

身份提供程序 (IdP)- 存储用户帐户的服务。在本例中，IdM 中持有的用户帐户将使用 RH-SSO 向 Keystone 呈现。

Service Provider (SP)- 需要 IdP 中用户进行身份验证的服务。在这种情况下，keystone 是赋予 IdM 用户仪表板访问权限的服务提供商。

您可以配置 Identity 服务(SP)与 RH-SSO (IdP)通信，后者也可以充当其他 IdP 的通用适配器。在此配置中，您可以将 keystone 指向 RH-SSO，RH-SSO 会将请求转发到它支持的身份提供程序（称为身份验证模块），它们目前包括 IdM 和 Active Directory。这可以通过使服务提供商(SP)和身份提供程序(IdP)交换元数据来实现，每个系统管理员都做出信任决定。结果是 IdP 可以自信地进行断言，SPM 可以接收这些断言。

其他资源

- 使用 Red Hat OpenStack Platform 和 Red Hat Single Sign-On 进行联邦