OpenStack Security Guide

[FAMILY Given]

havana (2014-03-17)

概要

This book provides best practices and conceptual information about securing an OpenStack cloud.

目次

図の一覧

Preface

Conventions

The OpenStack documentation uses several typesetting conventions:

Admonitions

Admonitions take three forms:

注記

This is a note. The information in a note is usually in the form of a handy tip or reminder.

重要

This is important. The information in an important admonition is something you must be aware of before moving on.

警告

This is a warning. The information in warnings is critical. Warnings provide additional information about risk of data loss or security issues.

Command prompts

Commands prefixed with the # prompt are to be executed by the root user. These examples can also be executed using the sudo command, if available.

Commands prefixed with the $ prompt can be executed by any user, including root.

ドキュメント変更履歴

このバージョンのガイドはすべての旧バージョンを置き換え、廃止します。以下の表はもっとも最近の変更点を記載しています。

Revision Date	Summary of Changes
December 2, 2013	• Chapter on Object Storage added.
October 17, 2013	• Havana リリース。
July 2, 2013	• Initial creation...

第1章 謝辞

OpenStack Security Group は、このドキュメントの作成を手助けしていただいた以下の組織の貢献に感謝いたします。

第2章 このドキュメントを作成した理由と方法

OpenStack が拡大を続け、製品が成熟してきたので、セキュリティが重要事項になってきました。OpenStack Security Group は包括的かつ権威のあるセキュリティガイドの必要性を認識しました。OpenStack セキュリティガイドは、OpenStack のセキュリティ向上を目的とした、セキュリティのベストプラクティス、ガイドライン、推奨事項の概要について記載しています。著者は さまざまな環境で OpenStack の導入やセキュア化をした専門知識をもたらします。

このガイドは OpenStack Operations Guide (OpenStack 運用ガイド) を補足します。既存の OpenStack 環境をセキュリティ強化するため、または OpenStack を用いたクラウド事業者のセキュリティ制御を評価するために参照できます。

目的

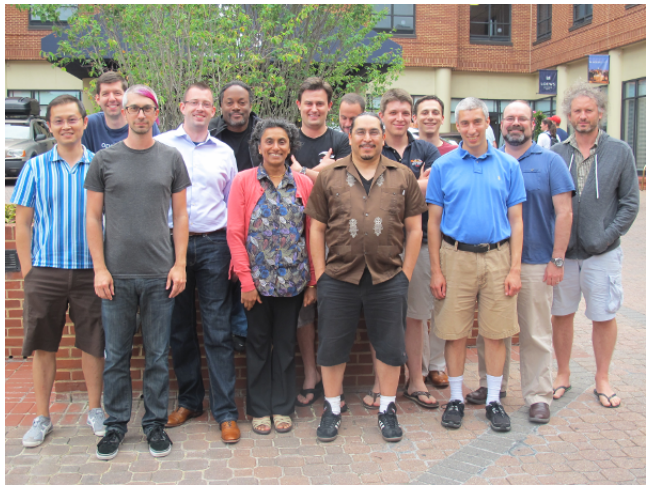
- OpenStack のセキュリティ領域を明確にする
- OpenStack をセキュア化するガイドを提供する
- 今日の OpenStack におけるセキュリティ考慮事項と実現可能な軽減策を強調する
- 将来のセキュリティ機能について議論する
- コミュニティ主導のナレッジ蓄積と普及を容易にする

執筆方法

OpenStack Operations Guide (OpenStack 運用ガイド) と同じく、Book Sprint メソッドを用いました。Book Sprint のプロセスにより、執筆作業の大部分を迅速に開発および作成できました。OpenStack Security Group のコーディネーターはファシリテーターとして Adam Hyde のサー

ビスを再び利用しました。企業サポートが得られ、オレゴン州ポートランドの OpenStack サミット中にプロジェクトが正式に公表されました。

チームは、グループの主要なメンバーが集まるために、メリーランド州アナポリスに集まりました。これは、公共部門のインテリジェンス・コミュニティーのメンバー、シリコンバレーのスタートアップ、いくつかの有名な大手技術企業の間での驚くべきコラボレーションです。Book Sprint は 2013 年 6 月の最終週に行われ、初版は 5 日間で作成されました。



チームは以下のとおりです。

- Bryan D. Payne, Nebula

  Dr. Bryan D. Payne は、Nebula の Security Research の Director です。また、OpenStack Security Group (OSSG) の共同創設者です。Nebula に参加する前は、Sandia National Labs、National Security Agency、BAE Systems、IBM Research に勤務していました。Georgia Tech College of Computing でシステムセキュリティを専攻し、コンピューターサイエンスの Ph.D. を取得しました。

- Robert Clark, HP

  Robert Clark は、Nebula の HP Cloud Services の Lead Security Architect です。また、OpenStack Security Group (OSSG) の共同創設者です。HP に入社する前は、UK Intelligence Community に勤務していました。脅威モデリング、セキュリティアーキテクチャー、仮想化技術に関する強固なバックグラウンドを持ちます。University of Wales のソフトウェアエンジニアリングの修士号を持っています。

- Keith Basil, Red Hat

  Keith Basil は Red Hat OpenStack の Principal Product Manager です。Red Hat の OpenStack 製品マネジメント、開発、戦略に注力しています。アメリカの公共部門の中で、アメリカの民間機関と委託業者向けの認定済み、セキュアかつハイパフォーマンスなクラウドアーキテクチャーの設計から、これまでの経験をもたらします。

- Cody Bunch, Rackspace

  Cody Bunch は Rackspace の Private Cloud architect です。『The OpenStack Cookbook』と VMware 自動化の書籍の共同執筆者です。

- Malini Bhandaru, Intel

  Malini Bhandaru は Intel のセキュリティアーキテクトです。Intel でプラットフォームの電力とパフォーマンス、Nuance でスピーチ製品、ComBrio でリモートモニタリングと管理、Verizon でウェブコマースに関するさまざまなバックグラウンドを持ちます。University of Massachusetts, Amherst で人工知能に関する Ph.D. を持っています。

- Gregg Tally, Johns Hopkins University Applied Physics Laboratory

  Gregg Tally は Asymmetric Operations Department の JHU/APL's Cyber Systems Group の Chief Engineer です。主にシステムセキュリティエンジニアリングに関する仕事をしています。以前は、サイバーセキュリティ研究プロジェクトに関わり、SPARTA、McAfee、Trusted Information Systems に勤務していました。

- Eric Lopez, VMware

  Eric Lopez is Senior Solution Architect at VMware's Networking and Security Business Unit where he helps customers implement OpenStack and VMware NSX (formerly known as Nicira's Network Virtualization Platform). Prior to joining VMware (through the company's acquisition of Nicira), he worked for Q1 Labs, Symantec, Vontu, and Brightmail. He has a B.S in Electrical Engineering/Computer Science and Nuclear Engineering from U.C. Berkeley and MBA from the University of San Francisco.

- Shawn Wells, Red Hat

Shawn Wells は Red Hat の Innovation Programs の Director です。アメリカ政府の中でオープンソース技術を適用、貢献、管理するプロセスを改善することに注力しています。さらに、SCAP Security Guide プロジェクトのアップストリームのメンテナーです。このプロジェクトは、 U.S. Military、NSA、DISA で仮想化とオペレーティングシステムの強化ポリシーを作成しています。NSA の契約者になる前は、大規模分散コンピューティング環境を利便化する SIGINT 収集システムを開発していました。

- Ben de Bont, HP

  Ben de Bont は HP Cloud Services の CSO です。その前は、MySpace の情報セキュリティグループ、MSN Security のインシデントレスポンスチームを率いていました。Queensland University of Technology のコンピューターサイエンスの修士号を保持しています。

- Nathanael Burton, National Security Agency

  Nathanael Burton は National Security Agency のコンピューターサイエンティストです。Agency に 10 年以上勤務し、分散システム、大規模ホスティング、オープンソースイニシアティブ、オペレーティングシステム、セキュリティ、ストレージ、仮想化技術に携わっています。Virginia Tech でコンピューターサイエンスの B.S. を取得しました。

- Vibha Fauver

  Vibha Fauver (GWEB, CISSP, PMP) は情報技術に関する 15 年以上の経験があります。専門分野はソフトウェアエンジニアリング、プロジェクト管理と情報セキュリティです。Computer & Information Science の B.S. と Engineering Management の M.S. を保持しています。Systems Engineering の資格を保持しています。

- Eric Windisch, Cloudscaling

  Eric Windisch は Cloudscaling の Principal Engineer です。OpenStack に 2 年以上貢献しています。ウェブホスティング業界における 10 年以上の経験から、ホスティング環境の分離性、テナント独立性の構築、インフラセキュリティに携わっています。2007 年以降、クラウドコンピューティング環境の構築と自動化に携わっています。

- Andrew Hay, CloudPassage

Andrew Hay は CloudPassage, Inc. の Applied Security Research の Director です。社内セキュリティおよび、ダイナミックパブリック、プライベート、ハイブリッドクラウドのホスティング環境向けに設計されたサーバーセキュリティ製品のセキュリティ研究チームを率いています。

- Adam Hyde

  Adam はこの Book Sprint をリードしました。彼は Book Sprint メソッドの創設者でもあり、一番経験豊富な Book Sprint のファシリテーターです。3000 人もの参加者がいる、フリーソフトウェアのフリーなマニュアルを作成するコミュニティである FLOSS Manuals の創設者です。また、Booktype の創設者でプロジェクトマネージャーです。 Booktype はオンラインで本の執筆、編集、出版を行うオープンソースプロジェクトです。

また、Book Sprint 期間中、Anne Gentle、Warren Wang、Paul McMillan、Brian Schott、Lorin Hochstein からの手助けがありました。

このドキュメントは、5日間の Book Sprint で作成されました。Book Sprint は、3〜5 日でドキュメントを作成するために、グループを集めて、強くコラボレーションし、プロセスをファシリテーションします。Adam Hyde により、設立され、開発された特別な方法でプロセスを強くファシリテーションします。詳細は Book Sprint のウェブページ http://www.booksprints.net を参照してください。

After initial publication, the following added new content:

- Rodney D. Beede, Seagate Technology

  Rodney D. Beede is the Cloud Security Engineer for Seagate Technology. He contributed the missing chapter on securing OpenStack Object Storage (Swift). He holds a M.S. in Computer Science from the University of Colorado.

このドキュメントへの貢献方法

このドキュメントの初期作業は、非常に空調の効いた部屋で行われました。ドキュメントスプリントの期間中、私たちグループのオフィスとして役に立ちました。

OpenStack ドキュメントに貢献する方法の詳細: http://wiki.openstack.org/Documentation/HowTo。

第3章 OpenStack の概要

本ガイドは、OpenStack のデプロイメントにおける、セキュリティに関する洞察を提供します。クラウドアーキテクト、デプロイ担当者、管理者などを対象読者としています。また、クラウドユーザーが知識を高めたり、プロバイダー選択に役立つ情報を記載している一方、監査担当者が、コンプライアンス認証関連の業務を支援する参考資料としてご利用いただくことができます。本ガイドは、クラウドのセキュリティに関心を持つ読者全般にもお奨めします。

OpenStack の各デプロイメントには、Linux ディストリビューション、データベースシステム、メッセージキュー、OpenStack のコンポーネント自体、アクセス制御ポリシー、ログサービス、セキュリティ監視ツールなどに及ぶ、多種多様なテクノロジーが採用されます。このため、デプロイに伴うセキュリティ問題が、同じように多様となることは当然です。それらの内容を奥深く分析するには、マニュアルが数冊必要となります。 本ガイドでは、OpenStack のセキュリティ問題とその対処方法を理解するために十分な情報を提供しつつ、さらなる情報の外部参照先を掲載することにより、バランスを図っています。本書は、全体を通読する方法または参考資料として必要箇所のみを参照する方法のいずれでもご利用いただくことができます。

本章では、プライベート、パブリック、ハイブリッドというクラウドの各種類について簡単に説明した後、後半に OpenStack のコンポーネントおよびそれらに関連するセキュリティ課題について概説します。

クラウドのタイプ

OpenStack は、クラウドテクノロジーの導入における重要なイネーブラーであり、一般的なデプロイメントユースケースがいくつかあります。これらは、パブリック、プライベート、およびハイブリッドモデルとして一般に知られています。以下のセクションでは、National Institute of Standards and Technology (NIST) のクラウドの定義 を取り上げ、OpenStack に適用するクラウドの異なるタイプについて説明します。

パブリッククラウド

NIST によると、パブリッククラウドは、一般市民が利用できるようにインフラストラクチャーが公開されているクラウドと定義されていま

す。OpenStack のパブリッククラウドは、通常サービスプロバイダーによって運用され、個人、法人、または料金を支払っている顧客が利用することができます。パブリッククラウドプロバイダーは、複数のインスタンスタイプに加えて、ソフトウェア定義ネットワーク、ブロックストレージなどの各種機能を公開することができます。パブリッククラウドはその性質上、より高いレベルのリスクにさらされます。パブリッククラウドの利用者は、選択したプロバイダーが必要な認定および認証を取得しているか、その他の法規制に関する考慮事項に対応しているかなどの点を確認しておく必要があります。パブリッククラウドプロバイダーは、ターゲット顧客に応じて、1 つまたは複数の法規制の影響を受ける場合があります。また、プロバイダーは、法規制の要件を満たす必要がない場合でも、管理インフラストラクチャーを外部の攻撃から保護するために、テナントの分離を確実に行う必要があります。

プライベートクラウド

At the opposite end of the spectrum is the private cloud. As NIST defines it, a private cloud is provisioned for exclusive use by a single organization comprising multiple consumers, such as business units. It may be owned, managed, and operated by the organization, a third-party, or some combination of them, and it may exist on or off premises. Private cloud use cases are diverse, as such, their individual security concerns vary.

コミュニティクラウド

NIST defines a community cloud as one whose infrastructure is provisioned for the exclusive use by a specific community of consumers from organizations that have shared concerns. For example, mission, security requirements, policy, and compliance considerations. It may be owned, managed, and operated by one or more of the organizations in the community, a third-party, or some combination of them, and it may exist on or off premises.

ハイブリッドクラウド

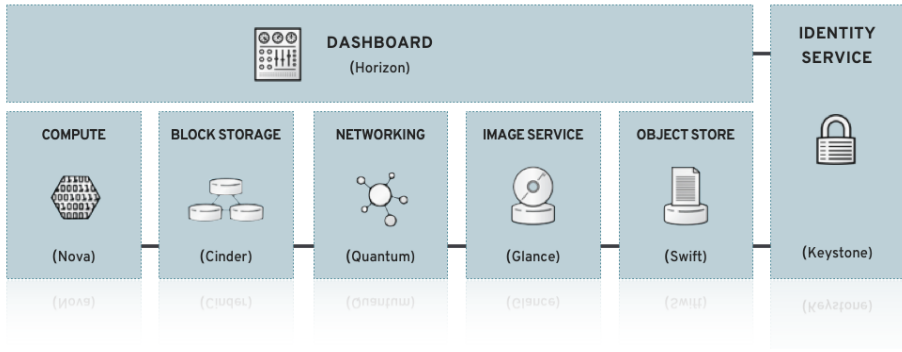
A hybrid cloud is defined by NIST as a composition of two or more distinct cloud infrastructures, such as private, community, or public, that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability, such as cloud bursting for load balancing between clouds. For example an online retailer may have their advertising and catalogue presented on a public cloud

that allows for elastic provisioning. This would enable them to handle seasonal loads in a flexible, cost-effective fashion. Once a customer begins to process their order, they are transferred to the more secure private cloud backend that is PCI compliant.

本ガイドにおいては、コミュニティクラウドとハイブリッドクラウドを同様に扱い、パブリッククラウドとプライベートクラウドの両極のみをセキュリティ面から明確に説明します。セキュリティ対策は、デプロイメントがプライベート/パブリッククラウドの連続体のどこに位置するかによって異なります。

OpenStack サービスの概観

OpenStack は、モジュール型アーキテクチャーを採用し、中核的な設計理念としてスケーラビリティと柔軟性を促進する一式のコアサービスを提供します。本章では、OpenStack のコンポーネントとそれらのユースケースおよびセキュリティに関する考慮事項を簡単に説明します。



コンピュート

OpenStack Compute Service (Nova) provides services to support the management of virtual machine instances at scale, instances that host multi-tiered applications, dev/test environments, "Big Data" crunching Hadoop clusters, and/or high performance computing.

The Compute Service facilitates this management through an abstraction layer that interfaces with supported hypervisors, which we address later on in more detail.

本ガイドの後半では、ハイパーバイザーと関連する仮想化スタックに焦点をあてて、包括的に解説します。

機能サポートの現在の状況に関する情報は、 OpenStack Hypervisor Support Matrix を参照してください。

The security of Compute is critical for an OpenStack deployment. Hardening techniques should include support for strong instance isolation, secure communication between Compute sub-components, and resiliency of public-facing API endpoints.

オブジェクトストレージ

The OpenStack Object Storage Service (Swift) provides support for storing and retrieving arbitrary data in the cloud. The Object Storage Service provides both a native API and an Amazon Web Services S3 compatible API. The service provides a high degree of resiliency through data replication and can handle petabytes of data.

オブジェクトストレージは、従来のファイルシステムストレージと異なる点を理解しておくことが重要です。メディアファイル（MP3、画像、ビデオ）や仮想マシンイメージ、バックアップファイルなどの静的データに使用するのに最適です。

オブジェクトのセキュリティは、アクセス制御と、伝送中および静止中のデータの暗号化に重点を置くべきです。その他の懸念事項には、システムの悪用、不法または悪意のあるコンテンツの保管、クロス認証の攻撃ベクトルなどに関する問題があげられます。

ブロックストレージ

The OpenStack Block Storage service (Cinder) provides persistent block storage for compute instances. The Block Storage Service is responsible for managing the life-cycle of block devices, from the creation and attachment of volumes to instances, to their release.

ブロックストレージのセキュリティ課題は、オブジェクトストレージの場合と同様です。

OpenStack Networking

The OpenStack Networking Service (Neutron, previously called Quantum) provides various networking services to cloud users (tenants) such as IP address management, DNS, DHCP, load balancing, and security groups (network access rules, like

firewall policies). It provides a framework for software defined networking (SDN) that allows for pluggable integration with various networking solutions.

OpenStack Networking により、クラウドテナントはゲストのネットワーク設定を管理することができます。ネットワークサービスに伴うセキュリティ上の問題には、 ネットワークトラフィックの隔離、可用性、完全性、機密性などがあげられます。

ダッシュボード

The OpenStack Dashboard Service (Horizon) provides a web-based interface for both cloud administrators and cloud tenants. Through this interface administrators and tenants can provision, manage, and monitor cloud resources. Horizon is commonly deployed in a public facing manner with all the usual security concerns of public web portals.

Identity サービス

The OpenStack Identity Service (Keystone) is a shared service that provides authentication and authorization services throughout the entire cloud infrastructure. The Identity Service has pluggable support for multiple forms of authentication.

ここでのセキュリティ課題には、認証の信頼、承認トークンの管理、セキュリティ保護された通信などがあげられます。

Image Service

The OpenStack Image Service (Glance) provides disk image management services. The Image Service provides image discovery, registration, and delivery services to Compute, the compute service, as needed.

前述したデータセキュリティに関する問題と同様に、ディスクイメージのライフサイクル管理には信頼されたプロセスが必要です。

その他の支援技術

OpenStack は、メッセージングに依存して、複数のサービス間の内部通信を行います。デフォルトでは、OpenStack は Advanced Message Queue Protocol (AMQP) をベースとするメッセージキューを使用します。これ

は、大半の OpenStack サービスと同様に、プラグ可能なコンポーネントをサポートしています。現在は、RabbitMQ、 Qpid、または ZeroMQ を実装バックエンドにすることができます。

メッセージキューシステムは、大半の管理コマンドが通過するので、OpenStack のデプロイメントにおける重要なセキュリティ課題です。メッセージキューのセキュリティについては、本ガイドの後半で詳述します。

一部のコンポーネントは、データベースを明示的に呼び出さずに使用します。データベースおよびそのコンテンツへのアクセスのセキュリティ保護は、もう一つのセキュリティ課題であるため、本ガイドの後半でさらに詳しく説明します。

第4章 セキュリティ境界と脅威

クラウドとは、セキュリティドメインと呼ばれる、機能やユーザー、共有セキュリティの関心事に基づいた論理コンポーネントの集まりであると要約できます。脅威に関するアクターやベクトルは、リソースへのアクセスや動機をベースに分類されます。OpenStack の目標は、リスクや脆弱性保護の目的にあわせてドメインごとにセキュリティの関心事についての判断材料を提供することです。

セキュリティドメイン

セキュリティドメインは、システム内の信頼性に関する共通の要件や期待を共有するユーザー、アプリケーション、サーバー、ネットワークのいずれかで構成されています。通常、これらのドメインには、同じ認証と承認（AuthN/Z）要件およびユーザーが指定されています。

これらのドメインをさらに分類する場合もありますが（該当箇所で説明）、一般的に OpenStack クラウドをセキュアにデプロイしていく上で最低限必要な部分を構成する、4 つの異なるセキュリティドメインのことを指します。以下に、これらのセキュリティドメインを示しています。

1. パブリック

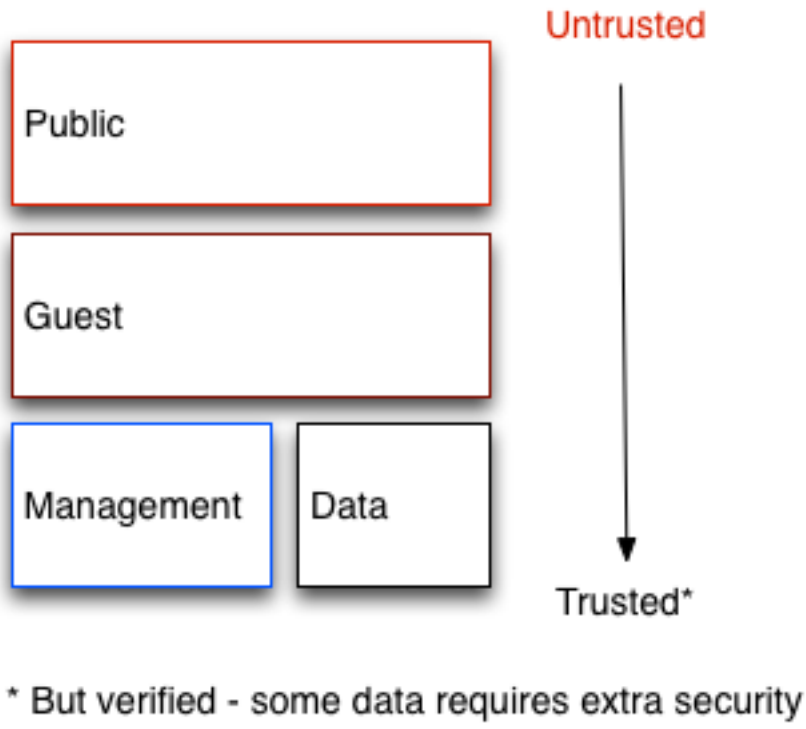
2. ゲスト

3. 管理

4. データ

上記のセキュリティドメインを選択したのは、個別にマッピング可能であること、または組み合わせると指定の OpenStack デプロイメントで存在する可能性のある信頼エリアの大部分を表すことができるためです。例えば、デプロイメントトポロジによっては、物理ネットワーク 1 つ vs 他のネットワークとなるように、ゲストとデータドメインの両方を組みわせて、ネットワークを物理的に分割するものもあります。いずれの場合も、クラウドオペレーターは、適切なセキュリティの関心事を認識

する必要があります。これらのドメインや信頼性に関する要件は、クラウドインスタンスがパブリック、プライベート、ハイブリッドのいずれであるかによって変わってきます。



パブリック

パブリックのセキュリティドメインとは、クラウドインフラストラクチャーの中で完全に Untrusted なエリアのことです。インターネット全体を指す場合や、単に権限を持たないネットワークを指す場合があります。機密性や完全性の要件を持つデータがこのドメインを通過する場合には、補完の制御を使用してこのデータを保護する必要があります。

このドメインは常に、 untrusted であると考えなければなりません。

ゲスト

ゲストのセキュリティドメインは、compute instance-to-instance トラフィックに通常使用されますが、API の呼び出しなどクラウドのオペ

レーションをサポートするサービスではなく、クラウド上のインスタンスが生成する compute データを処理します。

インスタンスの使用に関する厳密な制御がない、または制限なしに仮想マシンへインターネットアクセスが可能なパブリッククラウドのプロバイダーやプライベートクラウドのプロバイダーは、このドメインを untrusted であると見なすべきです。プライベートクライドプロバイダーは、インスタンスおよびすべてのテナントを確実に信頼できるように制御が設定されている場合のみ、このネットワークを内部、つまり trusted であると考えるようにしてください。

管理

管理セキュリティドメインは、サービスがやりとりをする場所です。このドメインは時に「制御プレーン」と呼ばれることもあり、このドメイン内のネットワークは設定パラメーター、ユーザー名、パスワードなどの機密データをトランスポートします。コマンドや制御トラフィックは通常このドメインに常駐し、完全性に関する強力な要件が必要となります。このドメインへのアクセスについては非常に制限されたものでなくてはならず、さらに監視も必要です。また、このセキュリティドメインでは、本ガイドで記載されているセキュリティのベストプラクティスすべてを採用するようにしてください。

多くのデプロイメントでは、この管理セキュリティドメインは trusted と考えられています。しかし、OpenStack のデプロイメントの場合、このドメインと他のものをブリッジするシステムが多数あるため、このドメインの信頼レベルは下がります。詳細は、「セキュリティドメインのブリッジ」 [17]を参照してください。

データ

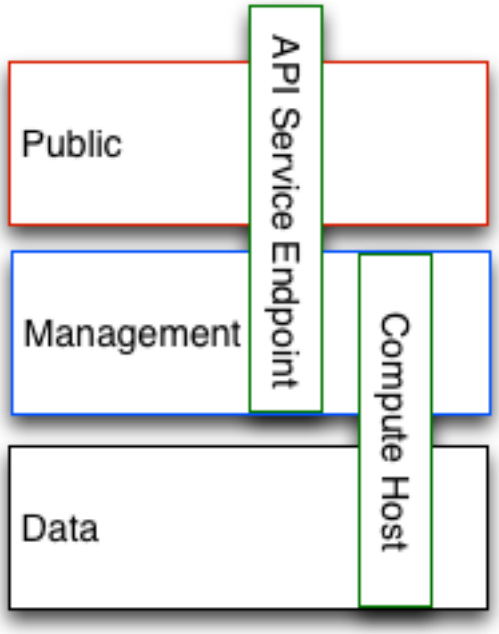
データセキュリティドメインは主に、OpenStack ではストレージサービスの情報に関係します。このネットワークを通過するデータの多くは、完全性や機密性に関する強力な要件を持ち、デプロイメントの種類によっては可用性に関する強力な要件が出てくる場合があります。

このネットワークの信頼レベルは、デプロイメントの意思決定により左右されるため、デフォルトの信頼レベルは割り当てていません。

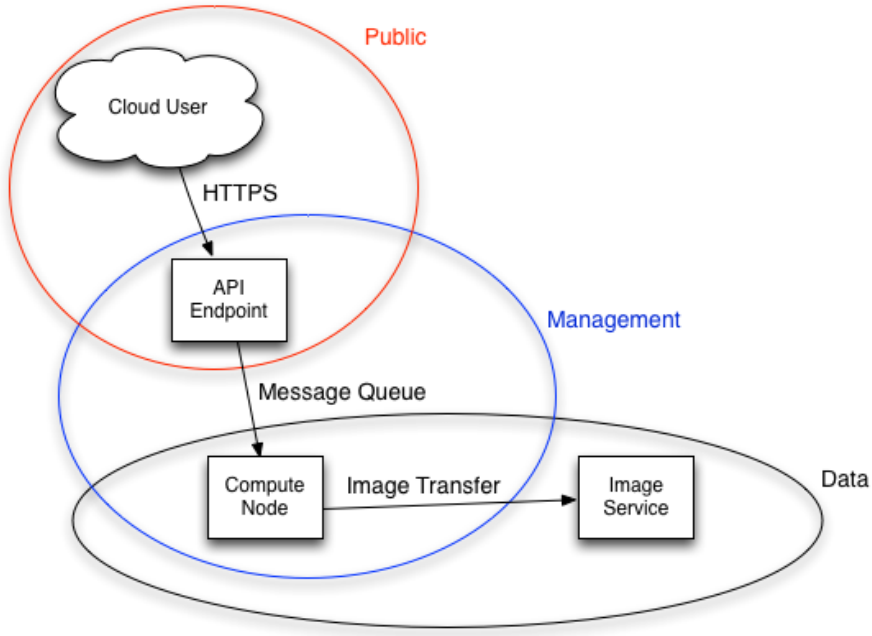
セキュリティドメインのブリッジ

ブリッジとは、複数のセキュリティドメイン内に存在するコンポーネントです。異なる信頼レベルまたは認証要件が指定されたセキュリテイド

メイン間をブリッジするコンポーネントは、慎重に設定する必要があります。ネットワークアーキテクチャーの中で、これらのブリッジは弱点となることが多くなっています。常に、ブリッジするドメインの中で最も高い信頼レベルのセキュリティ要件を満たすように、ブリッジを設定するようにしてください。多くの場合、攻撃の可能性の高さから、主にブリッジのセキュリティ制御について考慮する必要があります。



上記の図は、データドメインと管理ドメインをブリッジする compute ノードです。このように、compute ノードは管理ドメインのセキュリティ要件に見合うように設定する必要があります。同様に、この図の API エンドポイントは untrusted であるパブリックドメインと管理ドメインをブリッジしており、パブリックドメインから管理ドメインに伝搬しないように攻撃から保護されるように設定する必要があります。



デプロイ担当者は、ブリッジするどのドメインよりも高い基準でブリッジのセキュリティを確保するように考えるようにしてください。API エンドポイントの上記の例では、攻撃者はパブリックドメインから API エンドポイントをターゲットにして、情報漏洩や管理ドメインへアクセス権の獲得を期待しつつこのエンドポイントを利用するのです。

OpenStack のデザインではセキュリティドメインの分離が困難です。コアサービスは通常少なくとも 2 つのドメインをブリッジしているため、ドメインのセキュリティ制御を適用する場合、細心の注意を払う必要があります。

脅威の分類、アクター、攻撃ベクトル

クラウドデプロイメントの種類の多く（パブリックまたはプライベート）は、なんらかの攻撃にさらされています。本章では、攻撃者を分類して、各セキュリティドメインで考えられる攻撃の種類をまとめていきます。

脅威のアクター

脅威のアクターとは、防御の対象となりえる攻撃者のクラスを抽象的に表したものです。アクターの技術が高くなるにつれ、攻撃の軽減や防

止を成功させるために必要なセキュリティ制御にかかるコストが嵩みます。セキュリティはコスト、使いやすさ、防御の間でのトレードオフということになります。ここで記載した脅威のアクターすべてから、クラウドのデプロイメントを保護することはできません。OpenStack クラウドをデプロイする方は、デプロイメントと用途の間でバランスが確保できるポイントを決定する必要が出てきます。

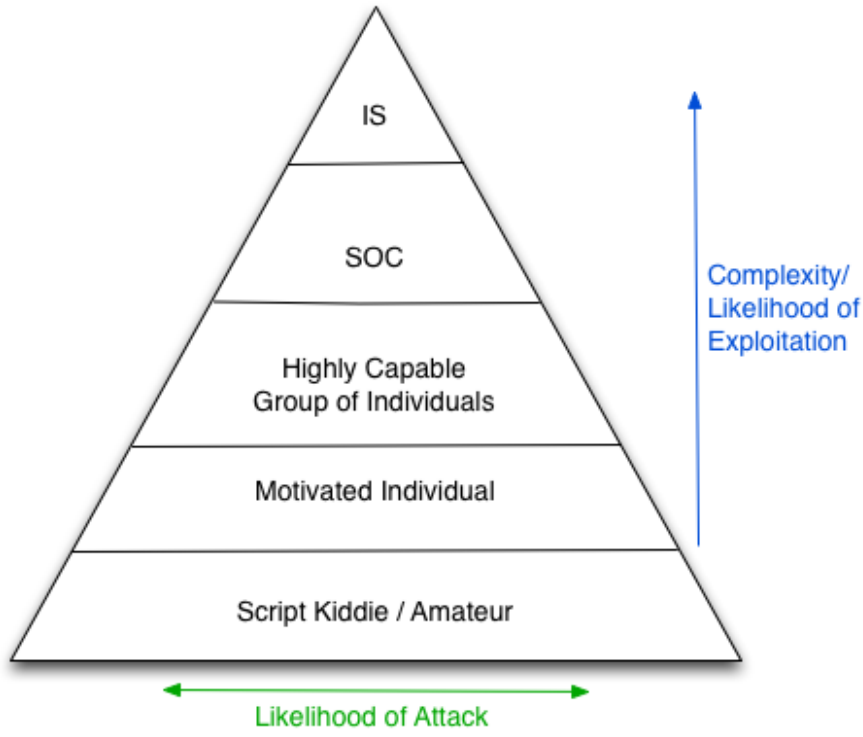
- インテリジェンスサービス — このガイドでは最も有能な攻撃者とされています。インテリジェンスサービスやその他の国家主体は、ターゲットに圧力をかけるために莫大なリソースを費やすことができます。他のどのアクターよりも能力があります。人、技術両方の面で非常に厳しい制御がないと、これらのアクターから防御することは極めて困難です。

- 重大組織犯罪 — 極めて有能で金銭で動く攻撃者グループ。エクスポロイト開発やターゲットのリサーチに対する資金を組織内で調達できます。最近、ロシアンビジネスネットワーク（RBN）などの組織が登場し、大規模なサイバー犯罪企業がサイバー攻撃がどのようにして商品として成り立ったかを証明しました。産業スパイ活動は、SOC グループに分類されます。

- 非常に有能な組織 — これは通常ビジネスから資金を調達しているのではありませんが、サービスプロバイダーやクラウドオペレーターに対して重大な脅威をもたらす可能性のある「ハクティビスト」タイプの組織のことを指します。

- 動機のある個人 — 一人で行動するこれらの攻撃者は、詐欺師または悪意のある従業員、不満を持った顧客、小規模の産業スパイなど多くのものに扮して攻撃します。

- スクリプトキディ — 自動化された脆弱性のスキャンやエクスプロイト。非標的型の攻撃。単なるいたずらの場合が多く、上記のアクターのいずれかによる情報漏洩により組織の評判に大きなリスクを与えます。



パブリック/プライベートの考慮点

通常プライベートクラウドは企業や組織により、内部のネットワークやファイアウォールの内側にデプロイされます。企業は、社内のネットワークから出すことのできるデータが何であるか、厳密な方針が設定されており、特定の目的ごとに別のクラウドを設定する場合さえもあります。プライベートクラウドのユーザーは通常、クラウドを所有して各自の行動に責任を課される組織内の従業員です。このような従業員は、クラウドにアクセスする前にトレーニングセッションに出席することもしばしばあり、定期的に予定されるセキュリティ認識トレーニングに参加する場合も多くあります。反対に、パブリッククラウドはユーザー、クラウドのユースケース、ユーザーの動機を断定することができません。このように、すぐにゲストのセキュリティドメインは、パブリッククラウドプロバイダーにとっては完全に untrusted な状態となります。

パブリッククラウドの攻撃対象領域での顕著な相違点は、サービスに対してインターネットアクセスを提供しなければならない点です。API エンドポイントやダッシュボードなど、インスタンスの接続性、インター

ネット経由でのファイルアクセス、クラウド制御のファブリックとの対話機能は、パブリッククラウドで必須アイテムなのです。

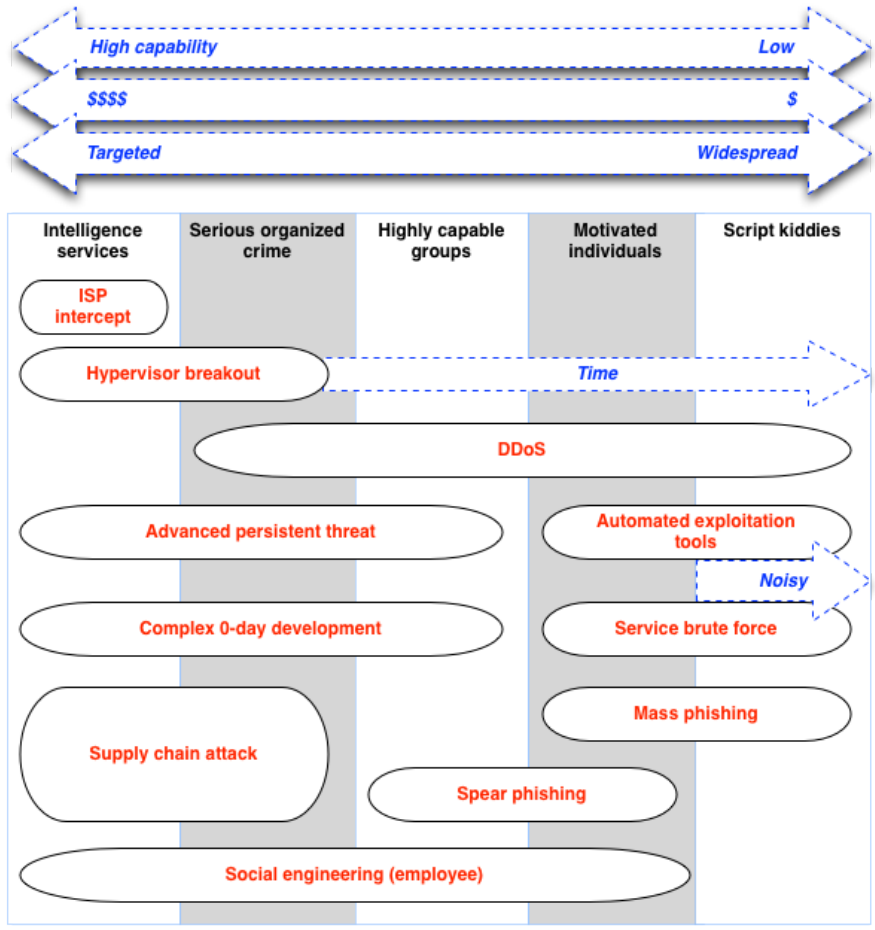
プライバシーの課題は、パブリッククラウドのユーザーとプライベートクラウドのユーザーとでは全く正反対になっています。プライベートクラウドで生成・格納されているデータは通常、データ損失防止（DLP）、ファイルの検査、ディープパケットインスペクション（DPI）、規範ファイアウォール（Prescriptive Firewall）などの技術をデプロイ可能なクラウドのオペレーターが所有します。反対に、パブリッククラウドには上記の様な制御の多くが存在しないため、プライバシーは、パブリッククラウドを採用する際の主な障害の 1 つとなっています。

アウトバウンド攻撃とレピュテーションリスク

クラウドデプロイメントからアウトバウンド方向で起こりえる不正使用に対して、十分な配慮が必要です。パブリックでも、プライベートでも、クラウドは多くのリソースが使用出来る状態になっている傾向にあります。ハッキングや与えられているアクセス権限（悪意のある従業員）のいずれかによりクラウド内に攻撃ポイントを設定した攻撃者は、これらのリソースにインターネット全体の負荷をかけることができます。コンピュートサービスがあるクラウドは、理想的な DDoS や総当り攻撃エンジンを作り出します。パブリッククラウドのユーザーは多くの場合、責任を負う必要がなく、自由に使用できるインスタンスをすぐにアウトバウンドの攻撃として作り出すことができるため、パブリッククラウドにとっては、この点はより差し迫った課題でしょう。悪意のあるソフトウェアをホストしたり、他のネットワークへ攻撃していたりしたことが判明すると、企業の評判に大きな打撃を与えることでしょう。防止の方法には、egress セキュリティグループ、アウトバウンドトラフィックの検査、顧客の教育・認識、詐欺や悪用軽減戦略などがあります。

攻撃の種類

以下の図は、前項で説明したアクターから出される可能性のある攻撃の種類を記載しています。このような図では常に例外が存在しますが、アクター毎に典型的であると考えられる攻撃の種類を一般論として記述しています。



攻撃の形式ごとの規範的な防御については、本書の対象範囲外となっています。上記の図は、対策を行うべき脅威の種類、脅威のアクターについて詳細な情報を得た状態で意思決定ができるように支援します。商業的なパブリッククラウドのデプロイに関しては重大な犯罪の防止などが含まれる場合があります。 政府で使用するプライベートクラウドをデプロイする方は、細心の注意を払って設置された対策施設やサプライチェーンなど、より厳密な保護メカニズムを設置する必要があります。反対に、基本的なデプロイメントやテスト環境を設定する方は、制御に関する制約が少なくて済むでしょう。

第5章 事例の概要

This guide refers to two running case studies, which are introduced here and referred to at the end of each chapter.

事例: プライベートクラウドビルダーのアリス

Alice deploys a private cloud for use by a government department in the US. The cloud must comply with relevant standards, such as FedRAMP. The security paperwork requirements for this cloud are very high. It must have no direct access to the internet: its API endpoints, compute instances, and other resources must be exposed to only systems within the department's network, which is entirely air-gapped from all other networks. The cloud can access other network services on the Organization's Intranet. For example, the authentication and logging services.

事例: パブリッククラウドプロバイダーのボブ

Bob is a lead architect for a company that deploys a large greenfield public cloud. This cloud provides IaaS for the masses and enables any consumer with a valid credit card access to utility computing and storage, but the primary focus is enterprise customers. Data privacy concerns are a big priority for Bob as they are seen as a major barrier to large-scale adoption of the cloud by organizations.

第6章 システムの文書化要件

OpenStack クラウドデプロイメントのシステム文書化は、その組織のエンタープライズ IT システムを対象とするテンプレートとベストプラクティスに従って行うべきです。組織には大抵、コンプライアンス要件が設定されており、それによって対象システムのインベントリ作成とアーキテクチャーの文書化を行う全体的なシステムセキュリティ計画が義務付けられている場合があります。動的なクラウドインフラストラクチャーを文書化し、情報を最新の状態に維持するのあたっては、業界全体の共通課題があります。

システムのロールとタイプ

The two broadly defined types of nodes that generally make up an OpenStack installation are:

- Infrastructure nodes. The nodes that run the cloud related services such as the OpenStack Identity Service, the message queuing service, storage, networking, and other services required to support the operation of the cloud.

- Compute, storage, or other resource nodes. Provide storage capacity or virtual machines for your cloud.

システムインベントリ

Documentation should provide a general description of the OpenStack environment and cover all systems used (production, development, test, etc.). Documenting system components, networks, services, and software often provides the bird's-eye view needed to thoroughly cover and consider security concerns, attack vectors and possible security domain bridging points. A system inventory may need to capture ephemeral resources such as virtual machines or virtual disk volumes that would otherwise be persistent resources in a traditional IT system.

ハードウェアインベントリ

Clouds without stringent compliance requirements for written documentation might benefit from having a Configuration Management Database (CMDB). CMDBs are normally used for hardware asset tracking and overall life-cycle management. By leveraging a CMDB, an organization can quickly identify cloud infrastructure hardware. For example, compute nodes, storage nodes, and network devices that exist on the network but that might not be adequately protected and/or forgotten. OpenStack provisioning system might provide some CMDB-like functions especially if auto-discovery features of hardware attributes are available.

ソフトウェアインベントリ

ハードウェアと同様に、OpenStack デプロイメント内のソフトウェアコンポーネントはすべて文書化しておくべきです。このコンポーネントには、システムデータベース、OpenStack ソフトウェアコンポーネントおよびサポートサブコンポーネント、ロードバランサー/リバースプロキシ/ネットワークアドレストランスレーターなどのサポートインフラストラクチャーソフトウェアなどが含まれます。このような信頼できる一覧を用意しておくことは、ソフトウェアの特定のクラスの侵害や脆弱性によってシステムが受ける全体的な影響を把握するために極めて重要となります。

ネットワークトポロジー

ネットワークトポロジーは、セキュリティドメイン間のデータフローとブリッジングポイントをはっきりと識別して強調するようにして作成すべきです。OpenStack の論理的なシステム境界とともに、ネットワークの受信および送信ポイントを明確にすることを推奨します。システムを完全に視覚的に網羅するには、図を複数作成する必要がある場合があります。また、ネットワークトポロジーの文書には、テナントに代わってシステムが作成した仮想ネットワークや、OpenStack によって作成された仮想マシンインスタンスとゲートウェイを含めるべきです。

サービス、プロトコル、ポート

The Service, Protocols and Ports table provides important additional detail of an OpenStack deployment. A table view of all services running within the cloud infrastructure can immediately inform, guide, and help check security procedures. Firewall

configuration, service port conflicts, security remediation areas, and compliance requirements become easier to manage when you have concise information available. Consider the following table:

Service	Protocols	Ports	Purpose	Used By	Security Domain(s)
beam.smp	AMQP	tcp/5672	AMQP message service	RabbitMQ	MGMT
tgtd	iSCSI	tcp/3260	iSCSI initiator service	iSCSI	PRIVATE (data network)
sshd	ssh	tcp/22	allows secure login to nodes and guest VMs	Various	MGMT, GUEST and PUBLIC as configured
mysqld	mysql	tcp/3306	MySQL database service	Various	MGMT
apache2	http	tcp/443	Horizon dashboard service	Tenants	PUBLIC
dnsmasq	dns	tcp/53	DNS services	Guest VMs	GUEST

サービス、プロトコル、ポートの表を参照すると、OpenStack のコンポーネント間の関係を理解するのに役立ちます。OpenStack のデプロイメントには、これと同様の情報を記録することを強く推奨します。

第7章 Case Studies: System Documentation

今回のケーススタディでは、アリスとボブがシステムの文書要件にどのように対処していくか見ていきます。上記で述べた文書には、ハードウェアおよびソフトウェア記録、ネットワーク図、システム設定の詳細などが含まれます。

アリスのプライベートクラウド

アリスは、FedRam 要件を満たす詳細文書が必要です。構成管理データベース（CMDB）を設定して、クラウド全体で使用されるハードウェア、ファームウェア、ソフトウェアバージョンの情報を格納していきます。また、セキュリティドメインや、複数のセキュリティドメインにまたがるサービスに細心の注意を払い、クラウドアーキテクチャーの詳細を示したネットワーク図も作成します。

アリスは、クラウドで実行中の各ネットワークサービス、バインド先のインターフェースやポート、各サービスに対するセキュリティドメイン、そのサービスが必要な理由を記録する必要があります。 Python Fabric ライブラリを使用して、セキュアシェル（SSH）でクラウド内の各システムにログインする自動化ツールを構築することにしました。このツールは、CMDB の情報を収集・格納して監査プロセスを簡素化します。

ボブのパブリッククラウド

今回のケーススタディでは、ボブはアリスと同様の手段を取ります。

第8章 管理の概要

クラウドデプロイメントは生きたシステムです。機械は老朽化して障害が発生し、ソフトウェアは古くなり、脆弱性が発見されます。設定にエラーや抜けがあった場合、ソフトウェアの修正を適用する必要が出た場合、セキュアかつ利便的に、これらの変更を加える必要があります。通常、これらの変更は構成管理などで解決されます。

同様に、悪意のある組織により設定または操作されないように、クラウドデプロイメントを保護することが重要です。コンピュートやネットワークの仮想化を採用するクラウド内の多くのシステムでは、OpenStackに適用される問題が明らかに存在し、整合性のライフサイクル管理で対応していく必要があります。

最後に、管理者は様々なオペレーション機能に対してクラウド上で指揮統制を行う必要があります。これらの指揮統制機能を理解、確保することが重要です。

第9章 継続的なシステム管理

クラウドには必ずバグがあります。その中にはセキュリティの問題も含まれています。このような理由から、セキュリティ更新や一般的なソフトウェア更新の適用準備を行うことが極めて重要です。例えば、構成管理ツールを賢く利用していくことになります。これについては以下で説明しています。また、更新が必要な時期を把握する必要があります。

脆弱性の管理

セキュリティ関連の変更に関するお知らせは、OpenStack Announce mailing list をサブスクライブしてください。セキュリティの通知は、パッケージ更新の一部としてサブスクライブしている可能性のある Linux ディストリビューションといったダウンストリームのパッケージでも掲載されます。

OpenStack のコンポーネントは、クラウドにあるソフトウェアのごく一部です。このような他のコンポーネントすべても最新の状態に保つことが重要です。データソースはそれぞれデプロイメント固有のものですが、主な目的はクラウド管理者は必要なメーリングリストにサブスクライブして関連のセキュリティ更新の通知を受信できるようにすることです。通常、Linux のアップストリームディストリビューションをトラッキングするのと同じくらいシンプルです。

注記

OpenStack は 2 つのチャネルからセキュリティ情報を発信しています。

- OpenStack セキュリティアドバイザリ (OSSA: OpenStack Security Advisories) は、OpenStack 脆弱性管理チーム (VMT: Vulnerability Management Team) が作成しています。コアとなる OpenStack サービスのセキュリティホールに関連するものです。VMT に関する詳細情報は、https://wiki.openstack.org/wiki/Vulnerability_Management を参照してください。

- OpenStack セキュリティノート（OSSN: OpenStack Security Notes）は、VMT の作業をサポートする OpenStack セキュリティグループ（OSSG: OpenStack Security Group）が 作成しています。OSSN はソフトウェアや一般的なデプロイメント設定のサポートにおける問題に対応しています。本書でも OSNN については全体的に参照しています。セキュリティノートは https://launchpad.net/ossn/ でアーカイブされています。

トリアージ

After you are notified of a security update, the next step is to determine how critical this update is to a given cloud deployment. In this case, it is useful to have a pre-defined policy. Existing vulnerability rating systems such as the common vulnerability scoring system (CVSS) v2 do not properly account for cloud deployments.

以下の例では、権限昇格、DoS（サービス妨害）、情報開示の 3 つのカテゴリーに脆弱性を分類した評価マトリクスを紹介しています。脆弱性の種類やインフラストラクチャー内での発生箇所を理解することで、裏付けに基いた対応意思決定を下すことができます。

Privilege Escalation describes the ability of a user to act with the privileges of some other user in a system, bypassing appropriate authorization checks. For example, a standard Linux user running code or performing an operation that allows them to conduct further operations with the privileges of the root users on the system.

サービス妨害（DoS）とは、サービスやシステムの中断を引き起こす脆弱性を悪用することを指します。これには、ネットワークリソースを大量に使用する分散型攻撃や、リソース割り当てのバグや誘導型でのシステム障害の問題などで一般的に引き起こされるシングルユーザー攻撃の両方が含まれます。

情報開示の脆弱性は、システムや操作の情報を公開します。これらの脆弱性は、情報開示のデバッグから認証情報やパスワードなどの重要なセキュリティデータの公開などが当てはまります。

	攻撃者の位置付け/権限レベル			
	外部	クラウドユーザー	クラウドの管理者	制御プレーン
権限昇格（3 つのレベル）	重要	なし	なし	なし

権限昇格（2つのレベル）	重要	重要	なし	なし
権限昇格（1つのレベル）	重要	重要	重要	なし
サービス妨害(DoS)	高	中	低	低
情報開示	重要/高	重要/高	中/低	低

This table illustrates a generic approach to measuring the impact of a vulnerability based on where it occurs in your deployment and the effect. For example, a single level privilege escalation on a Compute API node potentially allows a standard user of the API to escalate to have the same privileges as the root user on the node.

We suggest that cloud administrators use this table as a model to help define which actions to take for the various security levels. For example, a critical-level security update might require the cloud to be upgraded on a specified time line, whereas a low-level update might be more relaxed.

更新のテスト

You should test any update before you deploy it in a production environment. Typically this requires having a separate test cloud setup that first receives the update. This cloud should be as close to the production cloud as possible, in terms of software and hardware. Updates should be tested thoroughly in terms of performance impact, stability, application impact, and more. Especially important is to verify that the problem theoretically addressed by the update, such as a specific vulnerability, is actually fixed.

更新のデプロイ

更新の完全なテストが終了すると、実稼働環境にデプロイすることができます。このデプロイメントは、以下に記載の構成管理ツールで完全に自動的に行われます。

構成管理

実稼働環境の品質を持つクラウドは設定とデプロイメントの自動化ツールを必ず使用しています。こうすることで、人的ミスをなくし、クラウ

ドの迅速なスケールアウトが可能になります。自動化により、継続的した統合やテストが行いやすくなります。

OpenStack クラウドの構築時は、構成管理ツールまたはフレームワークを念頭に設計、実装に着手するように強く推奨します。構成管理により、OpenStack のように複雑なインフラストラクチャーの構築、管理、維持において陥りやすい多くの問題を回避することができます。構成管理ユーティリティに必要なマニフェスト、クックブック、テンプレートを作成することで、多くの文書や監督機関へのレポート要件を満たすことができます。さらに、構成管理は、BCP および DR プランの一部としても機能する可能性もあります。その場合、DR やセキュリティ侵害が合った場合にノードやサービスを既知の状態へ再構築することができます。

Additionally, when combined with a version control system such as Git or SVN, you can track changes to your environment over time and re-mediate unauthorized changes that may occur. For example, a nova.conf file or other configuration file falls out of compliance with your standard, your configuration management tool can revert or replace the file and bring your configuration back into a known state. Finally a configuration management tool can also be used to deploy updates; simplifying the security patch process. These tools have a broad range of capabilities that are useful in this space. The key point for securing your cloud is to choose a tool for configuration management and use it.

構成管理ソリューションは多数存在しますが、本書の作成時点で市場にあるソリューションで OpenStack 環境のサポートが強力なものは Chef と Puppet の 2 種類となっています。以下に完全ではありませんが、ツールのリストを示しています。

- Chef
- Puppet
- Salt Stack
- Ansible

ポリシーの変更

ポリシーや構成管理が変更されると、そのアクティビティをロギングして、新しいセットのコピーをバックアップすると慣習として良いでしょ

う。通常、このようなポリシーや設定は Git などのバージョン管理リポジトリに保存されています。

セキュアなバックアップとリカバリ

バックアップのプロシージャーとポリシーを全体的なシステムセキュリティプランに含めることは重要です。OpenStack のバックアップとリカバリー機能やプロシージャーについての適切な概要は、OpenStack 運用ガイドを参照してください。

セキュリティの課題

- 認証済みのユーザーおよびバックアップクライアントのみがバックアップサーバーにアクセスできるようにすること
- バックアップの移動やストレージにはデータ暗号化オプションを使用すること
- Use a dedicated and hardened backup servers. The logs for the backup server must be monitored daily and accessible by only few individuals.
- データのリカバリーオプションを定期的にテストすること。セキュアなバックアップからリストアが可能なものの 1 つにイメージがあります。情報漏洩などが発生した場合のベストプラクティスは、すぐに実行中のインスタンスを終了して、セキュアなバックアップリポジトリにあるイメージからインスタンスを再起動することです。

参考資料

- OpenStack 運用ガイド の バックアップとリカバリー
- http://www.sans.org/reading_room/whitepapers/backup/security-considerations-enterprise-level-backups_515
- OpenStack セキュリティ入門

セキュリティ監査ツール

セキュリティ監査ツールは、構成管理ツールを補完することができます。セキュリティ監査ツールは、セキュリティ制御の多くが指定のシステム設定を満たしていることを確認するプロセスを自動化します。これらのツールは、セキュリティ設定方針文書（例: STIG および NSA ガイ

ド）から個別のシステムインストール環境のギャップを埋めるサポートをします。例えば、SCAP は実行中のシステムと事前定義済みのプロファイルを比較することができます。SCAP はプロファイル内のどの制御に対応しているか、問題があるものはどれか、確認されていないものはどれかを詳細にまとめたレポートを出力します。

構成管理とセキュリティ監査ツールを組み合わせることで強力になります。監査ツールはデプロイメントの課題をハイライトし、構成管理ツールは各システムの変更プロセスを簡素化して監査の課題に対応していきます。このような方法で組み合わせて使用することで、これらのツールは、基本的なセキュリティの強化からコンプライアンスのバリデーションに至るまで、このようなセキュリティ要件を満たすクラウドを維持できるようにします。

Configuration management and security auditing tools will introduce another layer of complexity into the cloud. This complexity brings additional security concerns with it. We view this as an acceptable risk trade-off, given their security benefits. Securing the operational use of these tools is beyond the scope of this guide.

第10章 完全性ライフサイクル

We define integrity life cycle as a deliberate process that provides assurance that we are always running the expected software with the expected configurations throughout the cloud. This process begins with secure bootstrapping and is maintained through configuration management and security monitoring. This chapter provides recommendations on how to approach the integrity life-cycle process.

セキュアブートストラップ

クラウド内のノード（コンピュート、ストレージ、ネットワーク、サービス、およびハイブリッドのノードを含む）には、自動プロビジョニングプロセスを使用すべきです。このプロセスにより、ノードが一貫して正しくプロビジョニングされます。また、セキュリティパッチの適用、アップグレード、バグ修正、その他の重要な変更が円滑に行われます。このプロセスにより、クラウド内において最高権限で実行される新規ソフトウェアがインストールされるので、正しいソフトウェアがインストールされることを検証することが重要となります。これには、ブートプロセスの最初期段階が含まれます。

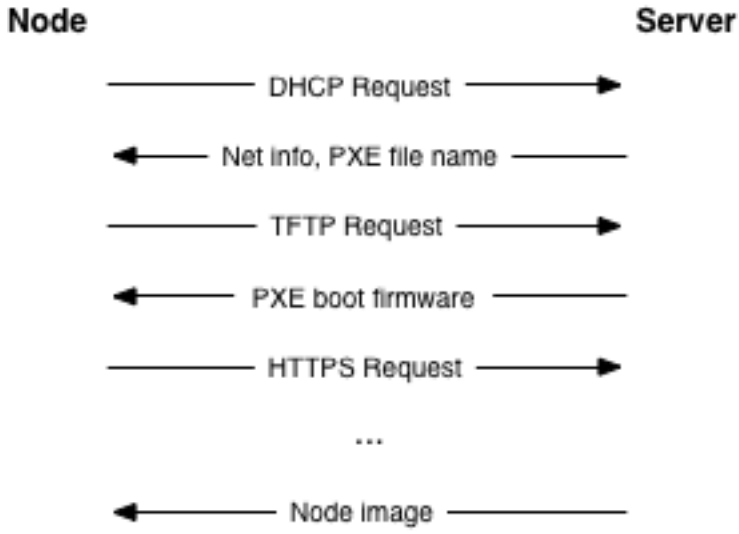
このような初期ブート段階の検証を可能にするさまざまな技術があります。通常は、Trusted Platform Module (TPM)、Intel Trusted Execution Technology (TXT)、Dynamic Root of Trust Measurement (DRTM)、Unified Extensible Firmware Interface (UEFI) などによるセキュアブートのハードウェアサポートが必要です。本ガイドでは、これらを総称してセキュアブートテクノロジーと呼びます。OpenStack ではセキュアブートの使用を推奨していますが、このデプロイに必要な諸作業には、各環境用にツールをカスタマイズするための高度の技術的スキルが必要である点を認識しています。セキュアブートの活用には、本ガイドに記載しているその他多くの推奨事項よりも深い統合とカスタマイズが必要になります。TPM テクノロジーはこの数年、大半のビジネスクラスのラップトップおよびデスクトップに通常搭載されていますが、BIOS のサポートとともにサーバーでも提供されるようになってきています。セキュアブートのデプロイには、適切な計画が不可欠です。

セキュアブートのデプロイに関する完全なチュートリアルは、本書の範囲外なので、その代わりとして、標準的なノードプロビジョニングプロセスにセキュアブートテクノロジーを統合する方法の枠組みを提供しま

す。クラウドアーキテクトが更に詳しい情報を確認するには、関連する仕様およびソフトウェア設定のマニュアルを参照することをお勧めします。

ノードのプロビジョニング

ノードは、プロビジョニングに Preboot eXecution Environment (PXE) を使用すべきです。これにより、ノードの再デプロイに必要な作業が大幅に軽減されます。標準的なプロセスでは、ノードがサーバーからさまざまなブート段階（実行するソフトウェアが徐々に複雑化）を受信する必要があります。



プロビジョニングには、管理セキュリティドメイン内の別個の分離したネットワークを使用することを推奨します。このネットワークは、上記に示した後続のブート段階のダウンロードに加えて、すべての PXE トラフィックを処理します。 ノードのブートプロセスは、安全性の低い DHCP および TFTP の 2 つの操作で開始する点に注意してください。次にブートプロセスは、ノードのデプロイに必要な残りの情報を SSL を介してダウンロードします。この情報には、initramfs とカーネルが含まれる場合があります。このプロセスは、ノードのデプロイに必要な残りの情報のダウンロードで終了します。これは、オペレーティングシステムのインストーラー、Chef または Puppet によって管理される基本インストール、またはディスクに直接書き込まれた完全なファイルシステムイメージの場合もあります。

While utilizing SSL during the PXE boot process is somewhat more challenging, common PXE firmware projects, such as iPXE, provide

this support. Typically this involves building the PXE firmware with knowledge of the allowed SSL certificate chain(s) so that it can properly validate the server certificate. This raises the bar for an attacker by limiting the number of insecure, plain text network operations.

検証済みブート

ブートプロセスの検証には、通常 2 つの異なる戦略があります。従来のセキュアブートは、プロセスの各ステップに実行されるコードを検証し、コードが正しくない場合にはブートを中止します。ブートアテステーションは、どのステップでどのコードが実行されるかを記録し、ブートプロセスが想定通りに完了した証拠として、この情報を別のマシンに提供します。いずれのケースにおいても、第 1 のステップでは、実行前にコードの各要素を計測します。この場合、計測値は実質的にはコードの SHA-1 ハッシュで、実行前に取得されます。 このハッシュは、TPM 内の Platform Configuration Register (PCR) に保管されます。

注記: ここで SHA-1 を使用するのは、TPM チップが対応しているためです。

各 TPM には少なくとも 24 の PCR が含まれます。TCG Generic Server Specification (v1.0、2005 年 3 月版) には、ブート時の完全性計測のための PCR の割り当てが定義されています。以下の表には、標準的な PCR 設定を記載しています。コンテキストには、その値がノードのハードウェア (ファームウェア) をベースに決定されるか、ノードにプロビジョニングされているソフトウェアをベースに決定されるかを示しています。一部の値は、ファームウェアのバージョンやディスクサイズ、その他の低レベルの情報によって影響を受けます。このため、設定管理の適切なプラクティスを整備し、デプロイするシステムが要望通りに設定されるようにしておくことが重要となります。

レジスター	計測の対象	コンテキスト
PCR-00	Core Root of Trust Measurement (CRTM)、BIOS コード、ホストプラットフォームの拡張機能	ハードウェア
PCR-01	ハードウェアプラットフォームの設定	ハードウェア
PCR-02	オプションの ROM コード	ハードウェア
PCR-03	オプションの ROM 設定およびデータ	ハードウェア

PCR-04	Initial Program Loader (IPL) Code. For example, master boot record.	ソフトウェア
PCR-05	IPL コードの設定およびデータ	ソフトウェア
PCR-06	状態遷移とウェイクイベント	ソフトウェア
PCR-07	ホストプラットフォームのメーカーによる制御	ソフトウェア
PCR-08	プラットフォーム固有、多くの場合はカーネル、カーネル拡張機能、ドライバー	ソフトウェア
PCR-09	プラットフォーム固有、多くの場合は Initramfs	ソフトウェア
PCR-10 から PCR-23	プラットフォーム固有	ソフトウェア

At the time of this writing, very few clouds are using secure boot technologies in a production environment. As a result, these technologies are still somewhat immature. We recommend planning carefully in terms of hardware selection. For example, ensure that you have a TPM and Intel TXT support. Then verify how the node hardware vendor populates the PCR values. For example, which values will be available for validation. Typically the PCR values listed under the software context in the table above are the ones that a cloud architect has direct control over. But even these may change as the software in the cloud is upgraded. Configuration management should be linked into the PCR policy engine to ensure that the validation is always up to date.

各メーカーは、サーバーの BIOS とファームウェアのコードを提供する必要があります。サーバー、ハイパーバイザー、オペレーティングシステムによって、事前設定される PCR 値の選択が異なります。実際のデプロイメントではほとんどの場合、既知の適切な量（「黄金の計測値」）と対照して各 PCR を検証することは不可能です。単一のベンダー の製品ラインの場合でも、一定の PCR の計測プロセスに一貫性がない場合があることが、経験により実証されています。各サーバーに基準値を定め、 PCR 値の予期せぬ変化を監視することを推奨します。選択したハイパーバイザーソリューションによっては、TPM プロビジョニングおよび監視プロセスを支援する サードパーティー製のソフトウェアが提供されている可能性があります。

The initial program loader (IPL) code will most likely be the PXE firmware, assuming the node deployment strategy outlined above. Therefore, the secure boot or boot attestation process can measure all of the early stage boot code, such as, bios, firmware, and the like, the PXE firmware, and the node kernel. Ensuring that each node has the correct versions of these pieces installed provides a solid foundation on which to build the rest of the node software stack.

Depending on the strategy selected, in the event of a failure the node will either fail to boot or it can report the failure back to another entity in the cloud. For secure boot, the node will fail to boot and a provisioning service within the management security domain must recognize this and log the event. For boot attestation, the node will already be running when the failure is detected. In this case the node should be immediately quarantined by disabling its network access. Then the event should be analyzed for the root cause. In either case, policy should dictate how to proceed after a failure. A cloud may automatically attempt to re-provision a node a certain number of times. Or it may immediately notify a cloud administrator to investigate the problem. The right policy here will be deployment and failure mode specific.

ノードのセキュリティ強化機能

この時点で、ノードが正しいカーネルと配下のコンポーネントでブートしていることが分かります。オペレーティングシステムのデプロイメントのセキュリティを強化するには、数多くの方法があります。これらの手順についての詳しい説明は本書の範囲外です。お使いのオペレーティングシステム固有のセキュリティ強化ガイドのアドバイスに従うことを推奨します。例えば、security technical implementation guides (STIG) や NSA guides を最初に参考にすると役立ちます。

ノードはその性質上、追加のセキュリティ強化が可能です。実稼働用のノードには、次の追加手順に従うことを推奨します。

- 可能な場合には、読み取り専用のファイルシステムを使用します。書き込みが可能なファイルシステムでは、実行が許可されないようにします。これは、/etc/fstab で指定するマウントオプションを使用して対処することが可能です。

- 強制アクセス制御ポリシーを使用して、インスタンス、ノードサービス、その他の重要なプロセスおよびノード上のデータが含まれるよう

にします。以下に記載の sVirt / SELinux および AppArmor についての説明を参照してください。

- 不要なソフトウェアパッケージは削除します。これにより、コンピュートノードの依存関係が比較的少なくなるので、インストールを小さく絞ることができます。

最後に、ノードのカーネルには、残りのノードが既知の良好な状態で起動することを検証するメカニズムを取り入れるべきです。これにより、ブート検証プロセスからシステム全体の検証に至るまでの必要なリンクが提供されます。手順はデプロイメントによって異なります。例えば、カーネルモジュールは、dm-verity を使用して、ファイルシステムをマウントする前に、そのファイルシステムを構成するブロック上のハッシュを検証することができます。

ランタイムの検証

ノードが稼働したら、長時間にわたって良好な状態で稼働を継続するように確保する必要があります。大まかに言うと、これには設定管理とセキュリティ監視が含まれます。これらの各領域の目標は異なります。両方を確認することにより、システムが希望通りに稼働していることをより確実に保証します。設定管理については、管理のセクションおよび次のセキュリティ監視で説明します。

侵入検知システム

ホストベースの侵入検知ツールは、クラウド内部の検証の自動化にも役立ちます。ホストベースの侵入検知ツールにはさまざまな種類があります。オープンソースで自由に利用できるツールもあれば、商用のツールもあります。通常、これらのツールは、さまざまなソースからデータを分析し、ルールセットやトレーニングに基づいてセキュリティ警告を出します。標準的な機能には、ログ解析、ファイルの完全性チェック、ポリシー監視、ルートキット検出などがあります。また、より高度なツール（カスタムの場合が多い）を使用すると、インメモリープロセスイメージがオンディスクの実行可能ファイルと一致するかどうかを確認して、実行中のプロセスの実行状態を検証することができます。

セキュリティ監視ツールの出力の処理方法は、クラウドアーキテクトにとっての重要なポリシー決定の一つです。オプションは実質的に 2 つあります。第 1 のオプションは、問題を調査して修正措置を取るように、人間に警告を発する方法です。これは、クラウド管理者向けのログまたはイベントのフィードにセキュリティ警告を組み込むことによって可能となります。第 2 のオプションは、イベントのログ記録に加えて、クラ

ウドが何らかの形の修復措置を自動的に実行するように設定する方法です。修復措置にはノードの再インストールから、マイナーなサービス設定の実行まで含めることができます。ただし、自動修復措置は、誤検知の可能性があるため、困難となる場合があります。

誤検知は、セキュリティ監視ツールが害のないイベントのセキュリティ警告を出した場合に発生します。セキュリティ警告ツールの性質上、時々誤検知が発生することは間違いありません。通常、クラウド管理者は、セキュリティ監視ツールを微調整して、誤検知を少なくすることができますが、これにより、全体的な検知率も同時に下がる場合があります。このような典型的トレードオフを理解し、クラウドにセキュリティ管理システムをセットアップする際には考慮に入れる必要があります。

ホストベースの侵入検知ツールの選択と設定はデプロイメントによって大幅に異なります。多様なホストベースの侵入検知/ファイル監視機能を実装する以下のオープンソースプロジェクトの検討から開始することをお勧めします。

- OSSEC
- Samhain
- Tripwire
- AIDE

ネットワーク侵入検知ツールは、ホストベースのツールを補完します。OpenStack には、特定のネットワーク IDS は組み込まれていませんが、OpenStack のネットワークコンポーネントである Neutron は、Neutron API を使用して異なるテクノロジーを有効にするプラグインメカニズムを提供しています。このプラグインのアーキテクチャーにより、テナントは API 拡張機能を開発して、ファイアウォール、侵入検知システム、仮想マシン間の VPN などの独自の高度なネットワークサービスを挿入/設定することができます。

ホストベースのツールと同様に、ネットワークベースの侵入検知ツールはデプロイメントによって異なります。 Snort は、先進的なオープンソースのネットワーク侵入検知ツールです。このツールを起点として、更に知識を深めてゆくとよいでしょう。

ネットワークおよびホストベースの侵入検知システムには、いくつかの重要なセキュリティ課題があります。

- It is important to consider the placement of the Network IDS on the cloud (for example, adding it to the network boundary and/or around sensitive networks). The placement depends on

your network environment but make sure to monitor the impact the IDS may have on your services depending on where you choose to add it. Encrypted traffic, such as SSL, cannot generally be inspected for content by a Network IDS. However, the Network IDS may still provide some benefit in identifying anomalous unencrypted traffic on the network.

- 一部のデプロイメントでは、ホストベースの IDS をセキュリティドメインブリッジ上の機密性の高いコンポーネントに追加する必要がある場合があります。ホストベースの IDS は、そのコンポーネント上の侵害された、あるいは許可されていないプロセスによる異常なアクティビティを検知することができます。IDS は管理ネットワーク上で警告およびログ情報を伝送すべきです。

第11章 管理インターフェース

管理者は、様々な運用機能に対してクラウドの管理統制を行う必要があります。また、これらの管理統制機能を理解して、セキュリティの確保を行うことが重要です。

OpenStack は、オペレーターやプロジェクト向けに複数の管理インターフェースを提供しています。

- OpenStack dashboard (Horizon)
- OpenStack API
- セキュアシェル (SSH)
- OpenStack Management Utilities (for example, nova-manage, glance-manage)
- 帯域外管理インターフェース (IPMI など)

ダッシュボード

The OpenStack dashboard (Horizon) provides administrators and tenants a web-based graphical interface to provision and access cloud-based resources. The dashboard communicates with the back-end services via calls to the OpenStack API (discussed above).

機能

- クラウド管理者として、ダッシュボードはクラウドのサイズや状態の俯瞰図を確認できます。また、ユーザーやプロジェクト (テナント) の作成、プロジェクト (テナント) へのユーザーの割り当て、ユーザーやプロジェクトで利用可能なリソースの制限設定が可能です。
- ダッシュボードでは、プロジェクト/ユーザーに対して、管理者が設定した制限値内で自身のリソースをプロビジョニングするためのセルフサービスポータルを提供します。

- The dashboard provides GUI support for routers and load-balancers. For example, the dashboard now implements all of the main Networking features.
- Hirozon は拡張可能な Django Web アプリケーションで、請求、監視、追加管理ツールなど、サードパーティーの製品やサービスを簡単にプラグインできるようにします。
- また、ダッシュボードはサービスプロバイダーや他の商業ベンダー向けにブランディングすることも可能です。

セキュリティの課題

- The dashboard requires cookies and JavaScript to be enabled in the web browser.
- The web server that hosts dashboard should be configured for SSL to ensure data is encrypted.
- バックエンドとの対話に使用する Horizon Web サービスおよび OpenStack API はいずれも、サービス妨害 (DoS) などの Web 攻撃ベクトルからの影響を受けるため、必ず監視が必要です。
- It is now possible (though there are numerous deployment/security implications) to upload an image file directly from a user's hard disk to OpenStack Image Service through the dashboard. For multi-GB images it is still strongly recommended that the upload be done using the Glance CLI
- ダッシュボードからセキュリティグループを作成・管理します。セキュリティグループにより、セキュリティポリシーの L3-L4 パケットをフィルダリングして仮想マシンの保護が可能になります。

参考資料

Grizzly リリースノート

OpenStack API

The OpenStack API is a RESTful web service endpoint to access, provision and automate cloud-based resources. Operators and users typically access the API through command-line utilities (for example, nova or glance), language-specific libraries, or third-party tools.

機能

- To the cloud administrator, the API provides an overall view of the size and state of the cloud deployment and allows the creation of users, tenants/projects, assigning users to tenants/projects, and specifying resource quotas on a per tenant/project basis.
- API はリソースのプロビジョニング、管理、アクセスに使用するプロジェクトインターフェースを提供します。

セキュリティの課題

- API サービスはデータが確実に暗号化されるように SSL の設定が必要です。
- Web サービスとして OpenStack API は、サービス妨害（DoS）攻撃など、よく知られている Web サイト攻撃ベクトルからの影響を受けます。

セキュアシェル（SSH）

Linux や Unix システムの管理にはセキュアシェル（SSH）を使用するのが業界の慣習となっています。SSH は通信にセキュアな暗号化プリミティブを使用します。一般的な OpenStack デプロイメントでの SSH の範囲や重要性において、SSH デプロイのベストプラクティスを把握することが重要です。

ホストキーのフィンガープリント

頻繁に見逃されるのが SSH ホストのキー管理の必要性です。OpenStack デプロイメントホストのすべてまたは多くが SSH サービスを提供します。このようなホストへの接続の信頼性を確保することが重要です。SSH ホストキーのフィンガープリントの検証に関して比較的セキュアでアクセス可能なメソッドを提供できないと、悪用やエクスプロイトの温床となるといっても過言ではありません。

SSH デーモンにはすべてプライベートのホストキーがあり、接続するとホストキーのフィンガープリントが提供されます。このホストキーのフィンガープリントは未署名のパブリックキーのハッシュです。これらのホストに SSH 接続する前に、ホストキーのフィンガープリントを把握しておくことが重要です。ホストキーのフィンガープリントの検証は中間者攻撃の検出に役立ちます。

通常、SSH デーモンがインストールされると、ホストキーが生成されます。ホストキーの生成時に、ホストには十分なエントロピーが必要になります。ホストキーの生成時にエントロピーが十分にないと、SSH セッションの傍受が発生してしまう可能性があります。

SSH ホストキーが生成されると、ホストキーのフィンガープリントはセキュアでクエリ可能な場所に保存されるはずです。特に有用なソリューションは、RFC-4255 で定義されていりょうに SSHFP リソースレコードを使用した DNS です。これをセキュアにするには、DNSSEC のデプロイが必要になります。

Management Utilities

The OpenStack Management Utilities are open-source Python command-line clients that make API calls. There is a client for each OpenStack service (for example, nova, glance). In addition to the standard CLI client, most of the services have a management command-line utility which makes direct calls to the database. These dedicated management utilities are slowly being deprecated.

セキュリティの課題

- 場合によっては専用の管理ユーテリティ（*-manage）は直接データベースへの接続を使用することがあります。
- 認証情報が含まれている .rc ファイルのセキュリティが確保されているようにします。

参考資料

OpenStack End User Guide section command-line clients overview

OpenStack エンドユーザーガイド の項 OpenStack RC ファイルのダウンロードとソース

帯域外管理インターフェース

OpenStack management relies on out-of-band management interfaces such as the IPMI protocol to access into nodes running OpenStack components. IPMI is a very popular specification to remotely manage, diagnose, and reboot servers whether the operating system is running or the system has crashed.

セキュリティの課題

- 強力なパスワードを使用してセーフガードするか、クライアント側の SSL 認証を使用してください。
- ネットワークインターフェースはプライベート（管理または個別）ネットワークに設定されていることw確認します。管理ドメインはファイアウォールか他のネットワークギアで分離してください。
- If you use a web interface to interact with the BMC/IPMI, always use the SSL interface, such as https or port 443. This SSL interface should NOT use self-signed certificates, as is often default, but should have trusted certificates using the correctly defined fully qualified domain names (FQDNs).
- Monitor the traffic on the management network. The anomalies might be easier to track than on the busier compute nodes.

また、帯域外管理インターフェースはグラフィカルのコンソールアクセスが可能な場合が多くあります。デフォルトではない可能性もありますが、これらのインターフェースは暗号化されていることがあります。これらのインターフェースの暗号化については、お使いのシステムのソフトウェア文書を確認してください。

参考資料

オフ状態のサーバーのハッキング

第12章 Case Studies: Management Interfaces

Previously we discussed typical OpenStack management interfaces and associated backplane issues. We will now approach these issues by returning to our Alice and Bob case study. Specifically, we will look into how both Alice and Bob will address:

- クラウド管理
- セルフサービス
- データの複製およびリカバリー
- SLA およびセキュリティの監視

アリスのプライベートクラウド

プライベートクラウドを構築する際、エアギャップはされていますが、アリスはサービス管理インターフェースを検討する必要があります。プライベートクラウドをデプロイする前に、システム文書を書き上げましあ。特に、どの OpenStack サービスが各セキュリティドメインに存在するかを特定しました。そこから、アリスは、IDS、SSL、暗号化、物理的なネットワークの分離を組み合わせてデプロイすることで、管理インターフェースへのアクセスをさらに制限しました。また、高可用性や冗長サービスも必要とするため、様々な OpenStack API サービスに対してインフラストラクチャーの冗長設定を行いました。

また、物理サーバーと Hypervisor は既知のセキュアな状態から十分に定義された設定へと確実に構築されるようにする必要があります。これを可能にするには、構成管理プラットフォームを合わせて使用して、準拠する必要のある規格や規定に従い各マシンを設定していきます。また、構成管理プラットフォームは、クラウドの状態を定期的に報告して、通常以外のことが発生した場合に既知の状態に修正することができます。さらに、PXE システムを使用することで、既知のベースイメージからノードを構築してハードウェア保証を提供することができます。ブートプロセス時に、そのハードウェアから提供される Intel TXT や関

連の信頼できるブート技術を有効にすることでさらなる保証を確保できます。

ボブのパブリッククラウド

パブリッククラウドのプロバイダーとして、ボブは管理インターフェースの継続的な可用性と、管理インターフェースへのトランザクションのセキュリティの両方を考慮しています。このように、ボブは、クラウドが実行するサービスに対して、複数の冗長 OpenStack API エンドポイントを実装します。さらに、パブリックネットワークでは、SSL を使用して、顧客とクラウドインターフェースの間のトランザクションをすべて暗号化します。クラウドの運用を分離するために、ボブは管理、インスタンス移行、ストレージネットワークを物理的に分離しました。

管理オーバーヘッドのスケーリングや削減を簡単にするため、構成管理システムを実装します。顧客のデータ保証に対しては、顧客ごとに要件が変わるためサービス商品としてバックアップを提供します。最後に、「ベアメタル」やノード全体のスケジュール機能を提供せず、管理オーバーヘッドの削減、運用効率の向上を図るため、ノードのブート時間におけるセキュリティ実装はありません。

第13章 SSL/TLSの導入

OpenStack のサービスは、管理ネットワーク経由の他の内部サービスからのリクエストと同様、パ ブリックネットワーク上のユーザによるリクエストを受信します。サービス間通信は、デプロイとアーキテクチャ選択によってはパブリックネットワーク経由で行われる事もあります。

パブリックネット上のデータはSecure Sockets Layer や Transport Layer Security (SSL/TLS)プロトコルのような暗号化方式を使用してセキュリティを確保すべきであるという事は一般に認識されている一方で、内部トラフィックの保護の為セキュリティドメイン分割に依存する事は不十分です。security-in-depth アプローチを用いて、管理ドメインサービスを含め、SSL/TLSを用いて全ドメインをセキュリティ確保する事を推奨します。テナントがVM分割を回避して、ハイパーバイザーやホストリソースへのアクセスを得て、APIエンドポイントやあらゆる他のサービスを妥協させる事は重大です。テナントが容易にインジェクトしたり、メッセージ・コマンド・その他クラウド上の管理機能に影響を与える又は制御する事が出来るようにスベきではありません。SSL/TLSは、OpenStackサービスへのユーザ通信やOpenStackサービス自体の相互間通信の認証、回避不能、秘密性、完全性を確保する仕組みを提供します。

Public Key Infrastructure (PKI)は認証、偽証不可、秘匿性、完全性を提供するセキュアなシステムを運用するハードウェア、ソフトウェア、ポリシーのセットです。PKIのコアコンポーネントは以下の通り。

- End Entity - 証明対象のユーザ、プロセス、システム
- 認証局 (Certification Authority、CA) - 証明ポリシーの定義、管理、証明書の発行
- Registration Authority (RA) -CAが一定の管理機能を委任する追加システム
- リポジトリ - End Entity が証明され、証明書の廃止リストが保存・参照される場所 - 時々「証明バンドル(Certificate bundle)」と呼ばれます。

- Relying Party - CAが有効であると証明するエンドポイント

PKIはデータと認証をセキュアにする暗号アルゴリズム、暗号モード(cipher mode)、プロトコルの フレームワークをバンドルしています。APIエンドポイントの為のSSL/TLS 使用を含み、Public Key Infrastructure (PKI)を用いて、全サービスをセキュアにする事をお勧めします。暗号化や通信路・メッセージの署名の為に、これら全ての問題を解決する事は重要です。プライベート証明と鍵の保護の為、ホスト自身がセキュアで、ポリシー、ネームスペース、その他の制御を実装しなければなりません。しかし、キー管理や保護のチャレンジはこれらの制御の必要性を削減したり、その重要性を失ったりはしません。

認証局(CA)

多くの組織には、内部のOpenStackユーザやサービス用に証明書を発行する為に使用されるべき場所用の自身の認証局(CA)、証明ポリシー、管理を備えたPublic Key Infrastructure (PKI)が設置されています 。加えて、パブリックセキュリティドメインがインターネットに面している所の組織は、幅広く認識された公共のCAにより署名された証明書が必要になるでしょう。管理ネットワーク上の暗号化通信用には、パブリックCAを使用しない事をお勧めします。代わりに、多くのデプロイでは自身の内部CAを設置していると思いますし、推奨します。

OpenStackクラウドアーキテクトには、内部のシステムと顧客が接するサービス用に、分断されたPKIデプロイの使用を検討する事をお勧めします。これは、クラウドをデプロイする人が他の物が内部のシステ ム用に証明書を要求・署名・デプロイする事を容易にするPKIインフラを制御できるようにします。異なる設定 は異なるセキュリティドメイン用にPKIデプロイを分割使用しても構いません。これは、デプロイする人が環境の暗号の分断を管理できるようにし、一方で発行された証明書が他方で認証されない事を保証します。

インターネットに面したクラウドのエンドポイント(あるいは証明書をバンドルした標準的なOS以外の何かがインストールされていると顧客が想定していない顧客インターフェース)上のSSL/TLSに対応に使用される証明書はOSの証明書バンドル中にインストールされるCAを用いてプロビジョニングされるべきです。通常、有名ベンダーにはベリサインやThawteを含みますが、他の多くのベンダーもあります。

証明書の作成・署名については多数の管理・ポリシー・技術的ハードルがあるため、証明書は、 ここで推奨されたガイドに加え、クラウドアーキテクトや運用者が工業リーダーやベンダのアドバイスを望みうる所です。

SSL/TLSライブラリ

OpenStackエコシステムやOpenStackが依存する様々なコンポーネント、サービス、アプリケーションはSSL/TLSライブラリを使用するよう実装され、設定ができるようになっています。OpenStack中のSSL/TLSとHTTPサービスは通常、非常にセキュアである事が証明され、FIPS 140-2用に検証されてきたOpenSSLを使用して実装されています。しかし、各アプリケーション又はサービスは、OpenSSLライブラリをどのように使用するかという点で、未だ脆弱性を招きうるという事を忘れないで下さい。

暗号化アルゴリズム、暗号モード、プロトコル

我々は TLS v1.1 又は v1.2 の使用のみ推奨します。SSL v3 と TLS v1.0 は互換性目的で使用出来ますが、我々は、注意深く、これらのプロトコルの有効化が強い要望としてある場合にのみ有効にする事をお勧めします。他のSSL/TLSバージョン(はっきり言えば古いバージョン)は使用すべきではありません。これらの古いバージョンには SSL v1 と v2 が含まれます。本書では暗号方式の初めから終わりまでの参考書を志向していない為、我々はあなたのOpenStackサービス中でどの特定アルゴリズムや暗号モードを有効・無効にすべきかについて指図する事を望みません。しかしながら、今後の情報としてお勧めしたい権威ある参考文献があります。

- National Security Agency, Suite B Cryptography
- OWASP Guide to Cryptography
- OWASP Transport Layer Protection Cheat Sheet
- SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements
- The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software
- OpenSSL and FIPS 140-2

概要

OpenStack コンポーネントの複雑さとデプロイの発展性を考慮すると、確実に各コンポーネントがSSL証明書・鍵・CAを適切に設定されている事

に注意を払う必要があります。以下のサービスは(標準機能又はSSLプロキシ経由可のどちらかで)SSLとPKIが利用可能な本書の後の章で議論します。

- Compute APIエンドポイント
- Identity APIエンドポイント
- Networking APIエンドポイント
- ストレージAPIエンドポイント
- メッセージングサーバー
- データベースサーバー
- ダッシュボード

本書の至る所で、我々はSSLをSSL/TLSプロトコルに関する推奨を示す略称として使用します。

第14章 Case Studies: PKI and Certificate Management

このケーススタディでは、アリスとボグがPKI認証局(CA)の構築と証明書管理をどのように行うのかについて解説します。

アリスのプライベートクラウド

アリスは政府機関のクラウドアーキテクトで、彼女の機関が独自のCAを運用している事を知っています。アリスは、彼女のPKIを管理して証明書を発行する職場の PKI オフィスにコンタクトします。アリスはこのCAによって発行された証明書を入手し、これらの証明書を使用するようパブリックと管理セキュリティドメインの両方のサービスを設定します。アリスの OpenStack デプロイが完全にインターネットから独立して存在するので、OpenStack サービスが彼女の組織の CA から発行されたクライアント証明書のみ許可するよう、外部のパブリックな CA プロバイダを含むデフォルトの全 CA バンドルが削除されている事を確認しています。

ボブのパブリッククラウド

ボブはパブリッククラウドのアーキテクトで、インターネットに接続された OpenStack サービスが主要な公的 CA から発行された証明書を確実に使用する必要があります。ボブは彼のパブリックな OpenStack サービス用の証明書を受領し、PKI と SSL を使用するようサービスを設定し、彼のサービス用の信用バンドル中に公的CAが含まれるようにします。更に、ボブはセキュリティ管理ドメイン内でサービス間の内部通信の更なる分断をしたいとも思っています。ボブは、彼の組織中で、内部CAを使用して彼の組織の PKI 管理と証明書の発行を担当しているチームにコンタクトします。ボブはこの内部CAが発行した証明書を入手し、これらの証明書を使用するよう管理セキュリティドメイン中での通信を行うサービスを設定し、内部CAが発行したクライアント証明書のみ許可するようサービスを設定します。

第15章 SSLプロキシとHTTPサービス

OpenStack エンドポイントはパブリックネットワーク上のエンドユーザと管理ネットワークを介して操作する同じデプロイ中の他 OpenStack サービスとの両方に対して API を提供する HTTP サービスです。これらのリクエスト（内部と外部の両方）を SSL 上で操作する事を強く推奨します。

API リクエストを SSL で暗号化する為に、APIサービスはSSLセッションを確立・切断するプロキシの後ろに位置する必要があります。下記の表はAPIリクエスト用にSSLトラフィックをプロキシ可能なソフトウェアサービスの（あまり厳密でない）一覧を示しています。

- Pound
- Stud
- nginx
- Apache httpd
- ハードウェアアプライアンス SSLアクセラレーションプロキシ

選択したSSLプロキシによって処理されるリクエストのサイズを気にする事は重要です。

例

Below we provide some sample recommended configuration settings for enabling SSL in some of the more popular web servers/SSL terminators. Note that we have SSL v3 enabled in some of these examples as this will be required in many deployments for client compatibility.

Before we delve into the configurations, we briefly discuss the ciphers' configuration element and its format. A more exhaustive

treatment on available ciphers and the OpenSSL cipher list format can be found at: ciphers.

```
ciphers = "HIGH:!RC4:!MD5:!aNULL:!eNULL:!EXP:!LOW:!MEDIUM"
```

or

```
ciphers = "kEECDH:kEDH:kRSA:HIGH:!RC4:!MD5:!aNULL:!eNULL:!EXP:!LOW:!MEDIUM"
```

Cipher string options are separated by ":", while "!" provides negation of the immediately following element. Element order indicates preference unless overridden by qualifiers such as HIGH. Let us take a closer look at the elements in the above sample strings.

kEECDH:kEDH	Ephemeral Elliptic Curve Diffie-Hellman (abbreviated as EECDH and ECDHE). Ephemeral Diffie-Hellman (abbreviated either as EDH or DHE) uses prime field groups. Both approaches provide Perfect Forward Secrecy (PFS). Ephemeral Elliptic Curves require the server to be configured with a named curve, and provide better security than prime field groups and at lower computational cost. However, prime field groups are more widely implemented, and thus typically both are included in list.
kRSA	Cipher suites using the RSA exchange, authentication or either respectively.
HIGH	Selects highest possible security cipher in the negotiation phase. These typically have keys of length 128 bits or longer.
!RC4	No RC4. RC4 has flaws in the context of TLS/SSL V3. See On the Security of RC4 in TLS and WPA.
!MD5	No MD5. MD5 is not collision resistant, and thus not acceptable for Message Authentication Codes (MAC) or signatures.
!aNULL:!eNULL	Disallows clear text

!EXP	Disallows export encryption algorithms, which by design tend to were weak, typically using 40 and 56 bit keys. US Export restrictions on cryptography systems have been lifted and no longer need to be supported.
!LOW:!MEDIUM	Disallows low (keys 56 or 64 bits long) and medium (128 bit long keys) ciphers because of their vulnerability to brute force attacks (example 2-DES). This constraint leaves acceptable Triple Data Encryption Standard (Triple DES) also known as Triple Data Encryption Algorithm (TDEA) and the Advanced Encryption Standard (AES), each of which has keys greater than equal to 128 bits and thus more secure.
Protocols	Protocols are enabled/disabled through SSL_CTX_set_options. We recommend disabling SSLv2 and enabling TLS or SSLv3 (which was standardised as TLS with a few changes).

Pound（AES-NI アクセラレーション付き）

```
## see pound(8) for details
daemon      1
######################################################################
## global options:
User        "swift"
Group       "swift"
#RootJail   "/chroot/pound"
## Logging: (goes to syslog by default)
##  0   no logging
##  1   normal
##  2   extended
##  3   Apache-style (common log format)
LogLevel    0
## turn on dynamic scaling (off by default)
# Dyn Scale 1
## check backend every X secs:
Alive       30
## client timeout
#Client     10
## allow 10 second proxy connect time
ConnTO      10
```

```
## use hardware-acceleration card supported by openssl(1):
SSLEngine   "aesni"
# poundctl control socket
Control "/var/run/pound/poundctl.socket"
######################################################################
## listen, redirect and ... to:
## redirect all swift requests on port 443 to local swift proxy
ListenHTTPS
    Address 0.0.0.0
    Port    443
    Cert    "/etc/pound/cert.pem"
    ## Certs to accept from clients
    ##  CAlist      "CA_file"
    ## Certs to use for client verification
    ##  VerifyList  "Verify_file"
    ## Request client cert - don't verify
    ##  Ciphers     "AES256-SHA"
    ## allow PUT and DELETE also (by default only GET, POST and HEAD)?:
    NoHTTPS11   0
    ## allow PUT and DELETE also (by default only GET, POST and HEAD)?:
    xHTTP       1
    Service
        BackEnd
            Address 127.0.0.1
            Port    80
        End
    End
End
```

Stud

この Stud の例は、クライアント互換性の為に SSL v3 を有効にしています。ciphers 行は必要に応じていじる事が出来ますが、しかしながらこの例の値は合理的な初期値です。

```
# SSL x509 certificate file.
pem-file = "
# SSL protocol.
ssl = on
# List of allowed SSL ciphers.
# OpenSSL's high-strength ciphers which require authentication
# NOTE: forbids clear text, use of RC4 or MD5 or LOW and MEDIUM strength
 ciphers
ciphers = "HIGH:!RC4:!MD5:!aNULL:!eNULL:!EXP:!LOW:!MEDIUM"
# Enforce server cipher list order
prefer-server-ciphers = on
# Number of worker processes
workers = 4
# Listen backlog size
```

```
backlog = 1000
# TCP socket keepalive interval in seconds
keepalive = 3600
# Chroot directory
chroot = ""
# Set uid after binding a socket
user = "www-data"
# Set gid after binding a socket
group = "www-data"
# Quiet execution, report only error messages
quiet = off
# Use syslog for logging
syslog = on
# Syslog facility to use
syslog-facility = "daemon"
# Run as daemon
daemon = off
# Report client address using SENDPROXY protocol for haproxy
# Disabling this until we upgrade to HAProxy 1.5
write-proxy = off
```

nginx

この nginx の例は、セキュリティを最大化する為に TLS v1.1 又は v1.2 を必要とします。ssl_ciphers 行は必要に応じていじる事ができますが、しかしながらこの例の値は合理的な初期値です。

```
server {
    listen : ssl;
    ssl_certificate ;
    ssl_certificate_key ;
    ssl_protocols TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!RC4:!MD5:!aNULL:!eNULL:!EXP:!LOW:!MEDIUM

    server_name _;
    keepalive_timeout 5;

    location / {

    }
}
```

Apache

```
<VirtualHost <ip address>:80>
  ServerName <site FQDN>
  RedirectPermanent / https://<site FQDN>/
```

```
</VirtualHost>
<VirtualHost <ip address>:443>
  ServerName <site FQDN>
  SSLEngine On
  SSLProtocol +SSLv3 +TLSv1 +TLSv1.1 +TLSv1.2,
  SSLCipherSuite HIGH:!RC4:!MD5:!aNULL:!eNULL:!EXP:!LOW:!MEDIUM
  SSLCertificateFile    /path/<site FQDN>.crt
  SSLCACertificateFile  /path/<site FQDN>.crt
  SSLCertificateKeyFile /path/<site FQDN>.key
  WSGIScriptAlias / <WSGI script location>
  WSGIDaemonProcess horizon user=<user> group=<group> processes=3 threads=10
  Alias /static <static files location>
  <Directory <WSGI dir>>
    # For http server 2.2 and earlier:
    Order allow,deny
    Allow from all

    # Or, in Apache http server 2.4 and later:
    # Require all granted
  </Directory>
</VirtualHost>
```

Apache2 中の Compute API SSL エンドポイント（短い WSGI スクリプトと組み合わせる必要あり）

```
<VirtualHost <ip address>:8447>
  ServerName <site FQDN>
  SSLEngine On
  SSLProtocol +SSLv3 +TLSv1 +TLSv1.1 +TLSv1.2,
  SSLCipherSuite HIGH:!RC4:!MD5:!aNULL:!eNULL:!EXP:!LOW:!MEDIUM
  SSLCertificateFile    /path/<site FQDN>.crt
  SSLCACertificateFile  /path/<site FQDN>.crt
  SSLCertificateKeyFile /path/<site FQDN>.key
  WSGIScriptAlias / <WSGI script location>
  WSGIDaemonProcess osapi user=<user> group=<group> processes=3 threads=10
  <Directory <WSGI dir>>
    # For http server 2.2 and earlier:
    Order allow,deny
    Allow from all

    # Or, in Apache http server 2.4 and later:
    # Require all granted
  </Directory>
</VirtualHost>
```

HTTP Strict Transport Security

全ての製品で HSTS を使用する事を推奨します。このヘッダは、ブラウザが単一のセキュアな接続を確立した後に、セキュアでない接続を確立

する事を防止します。パブリック上あるいは信用出来ないドメイン上の HTTP サービスをデプロイした場合、HSTS は特に重要です。HSTS を有効にするためには、全リクエストでこのようなヘッダを送信するよう Web サーバを設定します。

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

テストでは1日の短いタイムアウトで始め、テストでユーザに問題が発生しなかった事を確認した後で設定を１年まで増やします。一旦このヘッダに大きなタイムアウトを設定してしまうと、無効化する事は(設計上)非常に困難です。

第16章 APIエンドポイント構成に関する推奨事項

この章では外部と内部のエンドポイントのセキュリティ向上するための推奨事項を提供します。

内部API通信

OpenStackはパブリックとプライベート両方のAPIエンドポイントを提供します。デフォルトではOpenStackコンポーネントはパブリックとして定義されたエンドポイントを使用します。推奨はこれらのコンポーネントを適切なセキュリティドメイン内で使用するよう構成することです。

サービスはOpenStackサービスカタログに基づいて、それぞれのAPIエンドポイントを選択します。ここでの問題は、これらのサービスがリストされた外部もしくは内部APIエンドポイントの値に従わないことがあります。これは内部管理トラフィックが外部APIエンドポイントへルーティングされる可能性があります。

認証サービスのカタログ内の内部URL構成

The Identity Service catalog should be aware of your internal URLs. While this feature is not utilized by default, it may be leveraged through configuration. Additionally, it should be forward-compatible with expectant changes once this behavior becomes the default.

エンドポイント用の内部URL登録

```
$ keystone endpoint-create ¥
 --region RegionOne ¥
 --service-id=1ff4ece13c3e48d8a6461faebd9cd38f ¥
 --publicurl='https://public-ip:8776/v1/%(tenant_id)s' ¥
 --internalurl='https://management-ip:8776/v1/%(tenant_id)s' ¥
 --adminurl='https://management-ip:8776/v1/%(tenant_id)s'
```

内部URL用のアプリケーション構成

いくつかのサービスは特定のAPIエンドポイントの仕様を強制することができます。従って、それぞれのOpenStackサービスと他サービスとの通信は明示的に適切な内部APIエンドポイントへアクセスするよう構成する必要があります。

各プロジェクトで一貫性の無いAPIエンドポイントを提供しています。将来のリリースにおいてこれらの不一致を認証サービスカタログを使った一貫性で解決しようとしています。

構成例#1: Nova

```
[DEFAULT]
cinder_catalog_info='volume:cinder:internalURL'
glance_protocol='https'
neutron_url='https://neutron-host:9696'
neutron_admin_auth_url='https://neutron-host:9696'
s3_host='s3-host'
s3_use_ssl=True
```

構成例#2: Cinder

```
glance_host='https://glance-server'
```

Paste と ミドルウェア

OpenStack内のほぼ全てのAPIエンドポイントと他のHTTPサービスはPythonのPaste Deployライブラリを利用しています。これはアプリケーションの設定によってリクエストフィルターのパイプラインが操作が可能だと理解することがセキュリティの観点から重要になります。このパイプライン連鎖の中のそれぞれの要素はmiddlewareとして呼ばれています。パイプラインの中でフィルター順序を変更したり、ミドルウェアを追加すると予期しないセキュリティ上の影響が発生する可能性があります。

実装者がOpenStackの基本機能を拡張するためにミドルウェアを追加することは珍しくはありません。私たちは非標準のソフトウェアコンポーネントをHTTPリクエストパイプラインへ追加することによって生じる潜在的なセキュリティについて慎重に検討する事を推奨しています。

Paste Deployに関する追加情報 http://pythonpaste.org/deploy/

APIエンドポイントのプロセス分離とポリシー

特にパブリックなセキュリティドメインに属するAPIエンドポイントプロセスは可能な限り分離すべきです。ディプロイメント可能であれば、APIエンドポイントは分離のために増設されたホスト上に構成すべきです。

名前空間

多くのOSは現在コンパートメント化をサポートしています。Linuxではプロセスに独立したドメインを割り当てる名前空間をサポートしています。システムのコンパートメント化についてはこのマニュアルの別の部分で詳しく説明されています。

ネットワークポリシー

APIエンドポイントは一般的に複数のセキュリティドメインをまたがるため、APIプロセスのコンパートメント化には特別の注意を払うべきです。追加の情報に関してはこの章のSecurity Domain Bridging を参照してください。

慎重なデザインを行えば、ネットワークACLとIDS技術をネットワークサービス間の特定の通信に摘要する事が出来ます。重要なドメインをまたがるサービスとして、OpenStackのメッセージキューにこの手の明示的な強制は適しています。

ポリシーの強制はホストベースのファイアウォール(例えばiptables)やローカルポリシー(SELinuxやAppArmor)、グローバルなネットワークポリシーによって設定することができます。

強制アクセス制御

APIエンドポイントのプロセスはマシン上の他のプロセスと分離されるべきです。これらのプロセスの構成は任意のアクセス制御方法ではなく、強制アクセス制御によって制限されるべきです。これらのアクセス制御の目的はAPIエンドポイントのセキュリティ侵害の抑制と、特権侵害の防止です。強制アクセス制御を利用する事で、禁止されたリソースへのアクセスが厳しく制限され、早期の警告が得られるようになります。

第17章 Case Studies: API Endpoints

このケーススタディでは、アリスとボブがどうやってプライベートクラウドとパブリッククラウドのエンドポイント設定を堅牢化するかについて議論します。 アリスのプライベートクラウドは公開されたものではありませんが、不適切な使い方によるエンドポイント侵害を憂慮しています。ボブのパブリッククラウドは、外部からの攻撃に対してリスクを低減する措置を講じなければいけません。

アリスのプライベートクラウド

アリスが所属する組織では、パブリックとプライベートのエンドポイントへのアクセスに対してセキュリティ対策を講じることが義務付けられています。そこで彼女は、パブリックとプライベートのサービスに対して Apache SSL Proxy を構築しました。 また、アリスの組織では自前の認証局を用意しています。アリスは、公開鍵暗号基盤の管理と証明書を発行する部署からもらった証明書を、パブリック側と管理側のセキュリティドメイン両方に設定しました。 アリスの OpenStack 環境はインターネットからは完全に隔絶されているため、証明書から外部の公開認証局を含む CA バンドルを削除しました。これにより、アリスの OpenStack 環境が受け付ける証明書は、組織の認証局が発行したクライアント証明書のみになります。 アリスは内部アクセス用の Internal URL 越しに、全サービスを Keystone サービスカタログに登録し、また、ホストベースの侵入検知システムを全 API エンドポイントに設定しました。

ボブのパブリッククラウド

ボブもまた、パブリックとプライベートエンドポイントを守る必要があるため、 Apache SSL proxy をパブリックサービスと内部サービスの両方に使います。 パブリックサービス側には、よく知られている認証局が署名した証明書キーファイルを、内部サービス側には、自組織が発行した自己署名証明書を管理ネットワーク上のサービスに設定しました。サービスの登録は、内部アクセス用の Internal URL 越しに、Keystone サービスカタログに登録してあります。 また、ボブのパブリッククラウドサービスは、強制アクセス制御のポリシーで設定した SELinux 上

で動かしています。これにより万が一、公開サービスが攻撃されても、セキュリティ侵害による影響を減らすことができます。 さらに、ホストベースの侵入検知システムをエンドポイントに設定しました。

第18章 Identity

The OpenStack Identity Service (Keystone) supports multiple methods of authentication, including username & password, LDAP, and external authentication methods. Upon successful authentication, The Identity Service provides the user with an authorization token used for subsequent service requests.

Transport Layer Security TLS/SSL provides authentication between services and persons using X.509 certificates. Although the default mode for SSL is server-side only authentication, certificates may also be used for client authentication.

認証

無効なログイン試行

Identity Service は、ログイン試行が連続して失敗した後に、アカウントへのアクセスを制限する方法を提供していません。何度も失敗するログイン試行はブルートフォース攻撃（図「攻撃の種類」参照）のようなものです。これは、パブリッククラウドでは、より重要な問題です。

ログイン試行を指定した回数だけ失敗すると、アカウントをブロックするような外部認証システムを使用することにより、防止することができます。アカウントは、別の連絡手段を介してのみ、ロック解除するようにできます。

もし防止することが選択肢になければ、被害を減らすために、検知することができます。検知は、アカウントへの権限のないアクセスを特定するために、アクセス制御ログを頻繁にレビューすることを意味します。その他の改善法としては、ユーザーパスワードの強度のレビュー、ファイアウォールルールで攻撃のネットワーク送信元のブロックなどがあります。接続数を制限するという、Keystone サーバのファイアウォールルールは、攻撃の効率を悪くし、攻撃者をあきらめさせるために使用できます。

さらに、普通でないログイン回数や疑わしいアクションに対して、アカウントの活動状況を確認することは有用です。可能ならば、アカウントを無効化します。しばしば、このアプローチはクレジットカード提供者により、詐欺の検出や警告のために使用されます。

多要素認証

権限のあるユーザーアカウントにネットワークアクセス用の多要素認証を使用します。Identity Service はこの機能を提供できる Apache Web サーバーを通して外部認証サービスをサポートします。サーバーは証明書を使用したクライアント認証を強制することもできます。

この推奨事項は、管理者パスワードを流出させる可能性のある、ブルートフォース、ソーシャルエンジニアリング、標的型と無差別のフィッシング攻撃に対する防御になります。

認証方式

内部実装認証方式

Identity Service はユーザーのクレデンシャルを SQL データベースに保存できます。または、LDAP 対応のディレクトリサーバーを使用できます。Identity Service のデータベースは、保存されているクレデンシャルが漏洩するリスクを減らすために、他の OpenStack サービスが使用するデータベースと分離することもできます。

認証がユーザー名とパスワードで行われている場合、Identity Service は NIST Special Publication 800-118 (draft) により推奨されている、パスワード強度、有効期限、ログイン試行回数制限に関するポリシーを強制できません。より強固なパスワードポリシーを強制したい組織は、Keystone Identity Service 拡張や外部認証サービスの使用を検討すべきです。

LDAP により、組織の既存のディレクトリサービスやユーザーアカウント管理プロセスに Identity 認証を統合することをシンプルにできます。

OpenStack の認証と認可のポリシーは、外部 LDAP サーバーに権限委譲することができます。一般的なユースケースは、プライベートクラウドの導入を検討していて、すでに従業員とユーザーのデーターベースを持っている組織です。これは LDAP システムにあるかもしれません。権限のある認証のソースとして LDAP を使用することが、LDAP サービスに権限委譲している Identity Service に要求されます。このサービスが

ローカルに設定されたポリシーに基づいて認可または拒否します。トークンは認証が成功した場合に生成されます。

LDAP システムがユーザーに対して定義された、幹部社員、経理、人事などのような属性を持っている場合、これらはさまざまな OpenStack サービスにより使用するために Identity の中でロールとグループにマッピングされる必要があります。

Identity Service は OpenStack の外部にある認証用 LDAP サービスに書き込みを許可してはいけません。十分な権限を持つ keystone ユーザーが LDAP ディレクトリに変更を加えられるようになるからです。これにより、より広い範囲の組織に権限が増えたり、他の情報やリソースに権限のアクセスが容易になったりするかもしれません。このような環境では、ユーザーの払い出しが OpenStack 環境のレルムの範囲外になるかもしれません。

注記

keystone.conf のパーミッションに関する OpenStack Security Note (OSSN) があります。

There is an OpenStack Security Note (OSSN) regarding potential DoS attacks.

外部認証方式

組織は、既存の認証サービスとの互換性のために外部認証を実装したいかもしれません。または、より強固な認証ポリシー要件を強制するためかもしれません。パスワードが認証のもっとも一般的な形式ですが、キー入力ロギングやパスワード推測など、さまざまな方法で破られる可能性があります。外部認証サービスにより、弱いパスワードのリスクを最小化する他の認証形式を提供できます。

これらは以下のものが含まれます。

- パスワードポリシー強制: ユーザーパスワードが、長さ、文字種の量、有効期限、失敗試行回数の最低基準を満たしていることを要求します。
- Multi-factor authentication: The authentication service requires the user to provide information based on something they have, such as a one-time password token or X.509 certificate, and something they know, such as a password.
- Kerberos

認可

Identity Service はグループとロールの概念をサポートします。ユーザーはグループに所属します。グループはロールの一覧を持ちます。OpenStack サービスはユーザーがサービスにアクセスしようとしているロールを参照します。OpenStack ポリシー判定ミドルウェアにより、各リソースに関連付けられたポリシールール、ユーザーのグループとロール、テナント割り当てを考慮して、要求されたリソースへのアクセスが判断されます。

ポリシー強制ミドルウェアにより OpenStack リソースに細かなアクセス制御を実現できます。管理ユーザーのみが新しいユーザーを作成でき、さまざまな管理機能にアクセスできます。クラウドのテナントはインスタンスの稼動、ボリュームの接続などのみが実行できます。

公式なアクセス制御ポリシーの確立

ロール、グループ、ユーザーを設定する前に、OpenStack に必要なアクセス制御ポリシーをドキュメント化します。ポリシーは組織に対するあらゆる規制や法令の要求事項に沿っているべきです。アクセス制御設定のさらなる変更は公式なポリシーに従って実行されるべきです。ポリシーは、アカウントの作成、削除、無効化、有効化、および権限の割り当てに関する条件とプロセスを含めるべきです。定期的にポリシーをレビューし、設定が承認されたポリシーに従っていることを確認します。

サービス認可

OpenStack Cloud Administrator Guide に記載されているとおり、クラウド管理者は各サービスに対して Admin ロールを持つユーザーを定義する必要があります。このサービスユーザーアカウントは、サービスがユーザーを認証するための権限を提供します。

Nova と Swift のサービスは認証情報を保存するために ”tempAuth” ファイルと Identity Service を使用するよう設定できます。”tempAuth” ソリューションは、パスワードを平文で保存するため、本番環境で使用してはいけません。

The Identity Service supports client authentication for SSL which may be enabled. SSL client authentication provides an additional authentication factor, in addition to the username / password, that provides greater reliability on user identification. It reduces the risk of unauthorized access when user names and passwords may be compromised. However, there is additional

administrative overhead and cost to issue certificates to users that may not be feasible in every deployment.

注記

We recommend that you use client authentication with SSL for the authentication of services to the Identity Service.

クラウド管理者は権限のない変更から重要な設定ファイルを保護すべきです。これは SELinux のような強制アクセス制御のフレームワークで実現できます。これらには /etc/keystone.conf や X.509 証明書などがあります。

For client authentication with SSL, you need to issue certificates. These certificates can be signed by an external authority or by the cloud administrator. OpenStack services by default check the signatures of certificates and connections fail if the signature cannot be checked. If the administrator uses self-signed certificates, the check might need to be disabled. To disable these certificates, set insecure=False in the [filter:authtoken] section in the /etc/nova/api.paste.ini file. This setting also disables certificates for other components.

管理ユーザー

We recommend that admin users authenticate using Identity Service and an external authentication service that supports 2-factor authentication, such as a certificate. This reduces the risk from passwords that may be compromised. This recommendation is in compliance with NIST 800-53 IA-2(1) guidance in the use of multi factor authentication for network access to privileged accounts.

エンドユーザー

Identity Service は直接エンドユーザー認証を提供できます。または、組織のセキュリティポリシーや要求事項を確認するために外部認証方式を使用するよう設定できます。

ポリシー

各 OpenStack サービスは policy.json という JSON 形式のポリシーファイルを持ちます。ポリシーファイルはルールを指定します。ルー

ルは各リソースを決定します。リソースは API アクセスできます。ボリュームの接続やインスタンスの起動などです。

さまざまなリソースへのアクセス権をさらに制御するために、クラウド管理者がポリシーを更新できます。ミドルウェアによりさらにカスタマイズすることもできます。そのポリシーを参照しているグループやロールにユーザーを割り当てる必要があることに注意してください。

以下は Block Storage Service の policy.json ファイルの抜粋です。

```
{
   "context_is_admin":[
      [
         "role:admin"
      ]
   ],
   "admin_or_owner":[
      [
         "is_admin:True"
      ],
      [
         "project_id:%(project_id)s"
      ]
   ],
   "default":[
      [
         "rule:admin_or_owner"
      ]
   ],
   "admin_api":[
      [
         "is_admin:True"
      ]
   ],
   "volume:create":[

   ],
   "volume:get_all":[

   ],
   "volume:get_volume_metadata":[

   ],
   "volume:get_snapshot":[

   ],
   "volume:get_all_snapshots":[

   ],
   "volume_extension:types_manage":[
      [
```

```
        "rule:admin_api"
      ]
  ],
  "volume_extension:types_extra_specs":[
      [
        "rule:admin_api"
      ]
  ],
  "...":[
      [
        "...:..."
      ]
  ]
}
```

デフォルトのルールは、ユーザーが管理者であるか、ボリュームの所有者である必要があることを指定しています。つまり、ボリュームの所有者と管理者のみがボリュームを作成、削除、更新できます。ボリューム形式の管理など、他の特定の操作は管理ユーザーのみがアクセス可能です。

トークン

ユーザーが認証されると、トークンが生成され、認可とアクセスのために OpenStack で内部的に使用されます。デフォルトのトークンの有効期間は 24 時間です。この値はより短く設定することが推奨されますが、いくつかの内部サービスが処理を完了するために十分な時間が必要であるので注意する必要があります。トークンがすぐに失効すると、クラウドがサービスを提供できないかもしれません。これの例は、Compute Service がディスクイメージをハイパーバイザーのローカルキャッシュに転送するために必要な時間です。

The following example shows a PKI token. Note that, in practice, the token id value is about 3500 bytes. We shorten it in this example.

```
{
   "token":{
      "expires":"2013-06-26T16:52:50Z",
      "id":"MIIKXAY...",
      "issued_at":"2013-06-25T16:52:50.622502",
      "tenant":{
         "description":null,
         "enabled":true,
         "id":"912426c8f4c04fb0a07d2547b0704185",
         "name":"demo"
      }
   }
}
```

Note that the token is often passed within the structure of a larger context of an Identity Service response. These responses also provide a catalog of the various OpenStack services. Each service is listed with its name, access endpoints for internal, admin, and public access.

Identity Service はトークン失効をサポートします。これは、トークンを失効するため、失効済みトークンを一覧表示するために API として宣言されます。また、トークンをキャッシュしている各 OpenStack サービスが失効済みトークンを問い合わせるため、それらのキャッシュから失効済みトークンを削除するため、キャッシュした失効済みトークンの一覧に追加するためにもあります。

将来

ドメインはプロジェクト、ユーザー、グループの高いレベルでのコンテナーです。そのように、すべての Ketstone ベースの識別コンポーネントを一元的に管理するために使用されます。アカウントドメインを導入すると、サーバー、ストレージ、他のリソースは複数のプロジェクト（以前はテナントと呼ばれていました）の中で論理的にグループ化できます。これは、アカウントのようなマスターコンテナーの下でグループ化できます。さらに、複数のユーザーがアカウントドメインの中で管理でき、各プロジェクトで変化するロールを割り当てられます。

Keystone の V3 API はマルチドメインをサポートします。異なるドメインのユーザーは、異なる認証バックエンドで表現され、単一セットのロールと権限にマッピングされる異なる属性を持ちます。これらはさまざまなサービスリソースにアクセスするために、ポリシー定義で使用されます。

ルールにより管理ユーザーとテナントに所属するユーザーのみにアクセス権を設定されるかもしれないため、マッピングはささいなことである

かもしれません。他のシナリオの場合、クラウド管理者がテナントごとのマッピング作業を承認する必要があるかもしれません。

第19章 ダッシュボード

Horizon is the OpenStack dashboard that provides users a self-service portal to provision their own resources within the limits set by administrators. These include provisioning users, defining instance flavors, uploading VM images, managing networks, setting up security groups, starting instances, and accessing the instances via a console.

The dashboard is based on the Django web framework, therefore secure deployment practices for Django apply directly to Horizon. This guide provides a popular set of Django security recommendations, further information can be found by reading the Django deployment and security documentation.

The dashboard ships with reasonable default security settings, and has good deployment and configuration documentation.

基本的なウェブサーバーの設定

The dashboard should be deployed as a Web Services Gateway Interface (WSGI) application behind an HTTPS proxy such as Apache or nginx. If Apache is not already in use, we recommend nginx since it is lighter weight and easier to configure correctly.

When using nginx, we recommend gunicorn as the wsgi host with an appropriate number of synchronous workers. We strongly advise against deployments using fastcgi, scgi, or uWSGI. We strongly advise against the use of synthetic performance benchmarks when choosing a wsgi server.

When using Apache, we recommend mod_wsgi to host dashboard.

HTTPS

The dashboard should be deployed behind a secure HTTPS server using a valid, trusted certificate from a recognized certificate authority (CA). Private organization-issued certificates are only appropriate when the root of trust is pre-installed in all user browsers.

HTTP requests to the dashboard domain should be configured to redirect to the fully qualified HTTPS URL.

HTTP Strict Transport Security (HSTS)

HTTP Strict Transport Security (HSTS) を使用することが強く推奨されます。

NOTE: If you are using an HTTPS proxy in front of your web server, rather than using an HTTP server with HTTPS functionality, follow the Django documentation on modifying the SECURE_PROXY_SSL_HEADER variable.

HSTS の設定を含め、HTTPS の設定に関するより具体的な推奨事項とサーバー設定は、PKI/SSL の章全体を参照してください。

Front end Caching

Since dashboard is rendering dynamic content passed directly from OpenStack API requests, we do not recommend front end caching layers such as varnish. In Django, static media is directly served from Apache or nginx and already benefits from web host caching.

ドメイン名

Many organizations typically deploy web applications at subdomains of an overarching organization domain. It is natural for users to expect a domain of the form openstack.example.org. In this context, there are often many other applications deployed in the same second-level namespace, often serving user-controlled content. This name structure is convenient and simplifies name server maintenance.

We strongly recommend deploying horizon to a second-level domain, such as https://example.com, and advise against deploying horizon on a shared subdomain of any level, for example https://openstack.example.org or https://horizon.openstack.example.org. We also advise against deploying to bare internal domains like https://horizon/.

This recommendation is based on the limitations browser same-origin-policy. The recommendations in this guide cannot effectively protect users against known attacks if dashboard is deployed on a domain which also hosts user-generated content, such as scripts, images, or uploads of any kind, even if the user-generated content is on a different subdomain. This approach is used by most major web presences, such as googleusercontent.com, fbcdn.com, github.io, and twimg.com, to ensure that user generated content stays separate from cookies and security tokens.

Additionally, if you decline to follow this recommendation above about second-level domains, it is vital that you avoid the cookie backed session store and employ HTTP Strict Transport Security (HSTS). When deployed on a subdomain, dashboard's security is only as strong as the weakest application deployed on the same second-level domain.

静的メディア

Dashboard's static media should be deployed to a subdomain of the dashboard domain and served by the web server. The use of an external content delivery network (CDN) is also acceptable. This subdomain should not set cookies or serve user-provided content. The media should also be served with HTTPS.

Django media settings are documented at https://docs.djangoproject.com/en/1.5/ref/settings/#static-root.

Dashboard's default configuration uses django_compressor to compress and minify css and JavaScript content before serving it. This process should be statically done before deploying dashboard, rather than using the default in-request dynamic compression and copying the resulting files along with deployed code or to the CDN server. Compression should be done in a non-production build environment. If this is not practical, we recommend disabling resource compression entirely. Online compression dependencies (less, nodejs) should not be installed on production machines.

シークレットキー

Dashboard depends on a shared SECRET_KEY setting for some security functions. It should be a randomly generated string at least 64 characters long. It must be shared across all active Horizon instances. Compromise of this key may allow a remote attacker to execute arbitrary code. Rotating this key invalidates existing user sessions and caching. Do not commit this key to public repositories.

セッションバックエンド

Horizon の標準のセッションバックエンド (django.contrib.sessions.backends.signed_cookies) は、ブラウザに保存される、署名付きですが暗号化されていないクッキーにユーザーデータを保存します。この方法により、各 Horizon インスタンスがステートレスになるため、最も簡単なセッションバックエンドがスケールできるようになります。しかし、機微なアクセストークンをクライアントのブラウザーに保存し、それらをリクエストごとに送信するという犠牲を払うことになります。このバックエンドは、セッションデータが改ざんされていないことを保証しますが、データ自身は HTTPS で提供されるような暗号化以外には暗号化されていません。

If your architecture allows it, we recommend using django.contrib.sessions.backends.cache as your session backend with memcache as the cache. Memcache must not be exposed publicly, and should communicate over a secured private channel. If you choose to use the signed cookies backend, refer to the Django documentation understand the security trade-offs.

さらなる詳細は Django session backend documentation を参照してください。

許可されたホスト

Configure the ALLOWED_HOSTS setting with the domain or domains where Horizon is available. Failure to configure this setting (especially if not following the recommendation above regarding second level domains) opens Horizon to a number of serious attacks. Wild card domains should be avoided.

さらなる詳細は Django documentation on settings を参照してください。

クッキー

セッションクッキーは HTTPONLY に設定すべきです。

```
SESSION_COOKIE_HTTPONLY = True
```

Never configure CSRF or session cookies to have a wild card domain with a leading dot. Horizon's session and CSRF cookie should be secured when deployed with HTTPS:

```
Code CSRF_COOKIE_SECURE = True
SESSION_COOKIE_SECURE = True
```

パスワード自動補完

We recommend that implementers do not change the default password auto complete behavior. Users choose stronger passwords in environments that allow them to use the secure browser password manager. Organizations which forbid the browser password manager should enforce this policy at the desktop level.

クロスサイトリクエストフォージェリ (CSRF)

Django はcross-site request forgery (CSRF) 用の専用ミドルウェアを持ちます。

Dashboard is designed to discourage developers from introducing cross-site scripting vulnerabilities with custom dashboards. However, it is important to audit custom dashboards, especially ones that are javascript-heavy for inappropriate use of the @csrf_exempt decorator. Dashboards which do not follow these recommended security settings should be carefully evaluated before restrictions are relaxed.

クロスサイトスクリプティング (XSS)

Unlike many similar systems, OpenStack dashboard allows the entire Unicode character set in most fields. This means developers have less latitude to make escaping mistakes that open attack vectors for cross-site scripting (XSS).

Dashboard provides tools for developers to avoid creating XSS vulnerabilities, but they only work if developers use them correctly. Audit any custom dashboards, paying particular attention to use of the mark_safe function, use of is_safe with custom template tags, the safe template tag, anywhere auto escape is turned off, and any JavaScript which might evaluate improperly escaped data.

クロスオリジンリソースシェアリング (CORS)

ウェブブラウザが各レスポンスに限定的な CORS ヘッダーを付けて送信するよう設定します。Horizon のドメインとプロトコルのみを許可します。

```
Access-Control-Allow-Origin: https://example.com/
```

Never allow the wild card origin.

Horizon のイメージのアップロード

導入者はリソース枯渇とサービス妨害を防ぐ計画を実装していなければ、HORIZON_IMAGES_ALLOW_UPLOAD を無効化 することを強く推奨します。

アップグレード

Django security releases are generally well tested and aggressively backwards compatible. In almost all cases, new major releases of Django are also fully backwards compatible with previous releases. Dashboard implementers are strongly encouraged to run the latest stable release of Django with up-to-date security releases.

デバッグ

本番環境で DEBUG が False に設定されていることを確認します。Django では DEBUG により、あらゆる例外の発生時にスタックトレースと機微なウェブサーバーの状態情報が表示されます。

第20章 コンピュート

Compute Service (Nova) は最も複雑な OpenStack サービスの一つです。クラウドの隅々まで多くの場所で動作し、さまざまな内部サービスと通信します。この理由により、Compute Service 設定のベストプラクティスに関する推奨事項の多くは、本書を通して配布されます。管理、API エンドポイント、メッセージング、データベースのセクションで具体的な詳細を提供します。

仮想コンソールの選択

One decision a cloud architect will need to make regarding Compute Service configuration is whether to use VNC or SPICE. Below we provide some details on the differences between these options.

Virtual Network Computer (VNC)

OpenStack は Virtual Network Computer (VNC) プロトコルを使用して、プロジェクトと管理者がインスタンスのリモートデスクトップコンソールにアクセスできるように設定できます。

機能

- OpenStack Dashboard (Horizon) は HTML5 の非 VNC クライアントを使用して、ウェブページから直接インスタンスの VNC コンソールを提供できます。これには、nova-novncproxy サービスがパブリックネットワークから管理ネットワークにブリッジする必要があります。
- The nova command-line utility can return a URL for the VNC console for access by the nova Java VNC client. This requires the nova-xvpvncproxy service to bridge from the public network to the management network.

セキュリティの課題

- デフォルトのオープンなパブリックポートによる nova-novncproxy サービスと nova-xvpvncproxy サービスがトークン認証されます。

- デフォルトで、リモートデスクトップの通信は暗号化されません。Havana は Kerberos によりセキュア化された VNC 接続を実装することが期待されています。

参考資料

VNC ポートへのセキュアな接続

Simple Protocol for Independent Computing Environments (SPICE)

VNC の代替として、OpenStack は Simple Protocol for Independent Computing Environments (SPICE) プロトコルを使用した、仮想マシンへのリモートデスクトップアクセスを提供します。

機能

- SPICE は OpenStack Dashboard (Horizon) により直接インスタンスのウェブページでサポートされます。これには nova-spicehtml5proxy サービスが必要です。
- The nova command-line utility can return a URL for SPICE console for access by a SPICE-html client.

制限事項

- SPICE は VNC よりも多くの点で優れていますが、現在 spice-html5 ブラウザー統合は管理者がすべての利点を利用することができません。マルチモニター、USB パススルーなどの SPICE 機能の利点を利用するためには、管理ネットワークの中でスタンドアロン SPICE クライアントを使用することが推奨されます。

セキュリティの課題

- デフォルトのオープンなパブリックポートによる nova-spicehtml5proxy サービスがトークン認証されます。
- 機能と統合は進化中です。次のリリースの機能を確認し、推奨事項を作成します。
- VNC の場合のように、今のところ数人の利用者に制限して管理ネットワークから SPICE を使用することを推奨します。

参考資料

SPICE コンソール

Red Hat bug 913607

RDO Grizzly における SPICE のサポート

第21章 オブジェクトストレージ

OpenStack Object Storage (Swift) is a service that provides storage and retrieval of data over HTTP. Objects (blobs of data) are stored in an organizational hierarchy that offers anonymous read-only access or ACL defined access based on the authentication mechanism.

A consumer can store objects, modify them, or access them using the HTTP protocol and REST APIs. Backend components of Object Storage use different protocols for keeping the information synchronized in a redundant cluster of services. For more details on the API and the backend components see the OpenStack Storage documentation.

For this document the components will be grouped into the following primary groups:

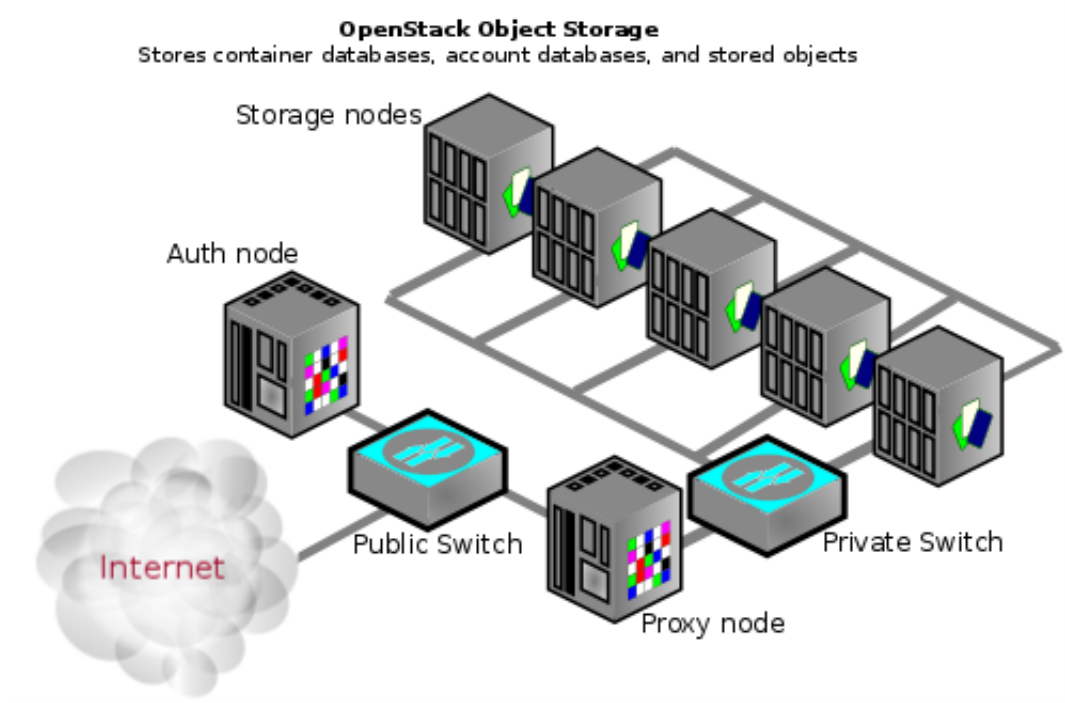
1. プロキシサービス

2. 認証サービス

3. Storage services

   - アカウントサービス
   - コンテナーサービス
   - オブジェクトサービス

図21.1 An example diagram from the OpenStack Object Storage Administration Guide (2013)



注記

An Object Storage environment does not have to necessarily be on the Internet and could also be a private cloud with the "Public Switch" being part of the organization's internal network infrastructure.

First thing to secure - the network

The first aspect of a secure architecture design for Object Storage is in the networking component. The Storage service nodes use rsync between each other for copying data to provide replication and high availability. In addition, the proxy service communicates with the Storage service when relaying data back and forth between the end-point client and the cloud environment.

注意

None of these use any type of encryption or authentication at this layer/tier.

This is why you see a ”Private Switch” or private network ([V]LAN) in architecture diagrams. This data domain should be separate from other OpenStack data networks as well. For further discussion on security domains please see 4章セキュリティ境界と脅威 [15].

ヒント

Rule: Use a private (V)LAN network segment for your Storage services in the data domain.

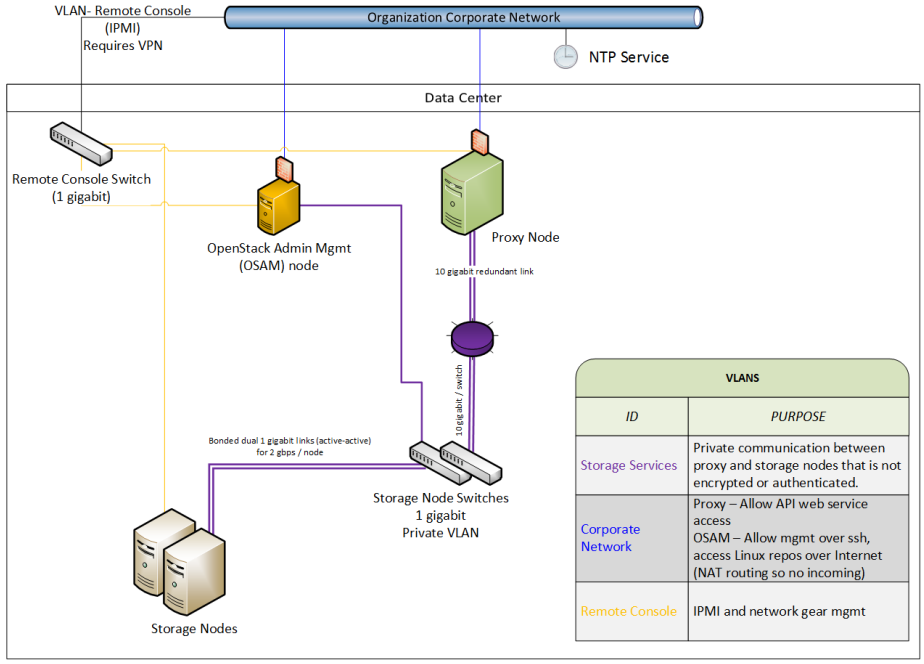
This necessitates that the Proxy service nodes have dual interfaces (physical or virtual):

1. One as a ”public” interface for consumers to reach

2. Another as a ”private” interface with access to the storage nodes

The following figure demonstrates one possible network architecture.

図21.2 Object storage network architecture with a management node (OSAM)



Securing services - general

Service runas user

It is recommended that you configure each service to run under a non-root (UID 0) service account. One recommendation is the username "swift" with primary group "swift."

File permissions

/etc/swift contains information about the ring topology and environment configuration. The following permissions are recommended:

```
#chown -R root:swift /etc/swift/*
#find /etc/swift/ -type f -exec chmod 640 {} ¥;
#find /etc/swift/ -type d -exec chmod 750 {} ¥;
```

This restricts only root to be able to modify configuration files while allowing the services to read them via their group membership in "swift."

Securing storage services

The following are the default listening ports for the various storage services:

Service Name	ポート	種別
アカウントサービス	6002	TCP
コンテナーサービス	6001	TCP
オブジェクトサービス	6000	TCP
Rsync	873	TCP

Authentication does not happen at this level in Object Storage. If someone was able to connect to a Storage service node on one of these ports they could access or modify data without authentication. In order to secure against this issue you should follow the recommendations given previously about using a private storage network.

Object storage "account" terminology

An Object Storage "Account" is not a user account or credential. The following explains the relations:

OpenStack Object Storage Account	Collection of containers; not user accounts or authentication. Which users are associated with the account and how they may access it depends on the authentication system used. See authentication systems later. Referred to in this document as OSSAccount.
OpenStack Object Storage Containers	Collection of objects. Metadata on the container is available for ACLs. The meaning of ACLs is dependent on the authentication system used.
OpenStack Object Storage Objects	The actual data objects. ACLs at the object level are also possible with metadata. It is dependent on the authentication system used.

ヒント

Another way of thinking about the above would be: A single shelf (Account) holds zero or more -> buckets

(Containers) which each hold zero or more -> objects. A garage (Object Storage cloud environment) may have multiple shelves (Accounts) with each shelf belonging to zero or more users.

At each level you may have ACLs that dictate who has what type of access. ACLs are interpreted based on what authentication system is in use. The two most common types of authentication providers used are Keystone and SWAuth. Custom authentication providers are also possible. Please see the Object Storage Authentication section for more information.

Securing proxy services

A Proxy service node should have at least two interfaces (physical or virtual): one public and one private. The public interface may be protected via firewalls or service binding. The public facing service is an HTTP web server that processes end-point client requests, authenticates them, and performs the appropriate action. The private interface does not require any listening services but is instead used to establish outgoing connections to storage service nodes on the private storage network.

SSL/TLS の使用

The built-in or included web server that comes with Swift supports SSL, but it does not support transmission of the entire SSL certificate chain. This causes issues when you use a third party trusted and signed certificate, such as Verisign, for your cloud. The current work around is to not use the built-in web server but an alternative web server instead that supports sending both the public server certificate as well as the CA signing authorities intermediate certificate(s). This allows for end-point clients that have the CA root certificate in their trust store to be able to successfully validate your cloud environment's SSL certificate and chain. An example of how to do this with mod_wsgi and Apache is given below. Also consult the Apache Deployment Guide

```
# apt-get install libapache2-mod-wsgi
```

Modify file /etc/apache2/envvars with

```
export APACHE_RUN_USER=swift
export APACHE_RUN_GROUP=swift
```

An alternative is to modify your Apache conf file with

```
User swift
Group swift
```

Create a swift directory in your Apache document root:

```
# mkdir /var/www/swift/
```

Create the file $YOUR_APACHE_DOC_ROOT/swift/proxy-server.wsgi:

```
from swift.common.wsgi import init_request_processor application, conf,
 logger, log_name = ¥
            init_request_processor('/etc/swift/proxy-server.conf','proxy-
server')
```

HTTP リッスンポート

You should run your Proxy service web server as a non-root (no UID 0) user such as "swift" mentioned before. The use of a port greater than 1024 is required to make this easy and avoid running any part of the web container as root. Doing so is not a burden as end-point clients are not typically going to type in the URL manually into a web browser to browse around in the object storage. Additionally, for clients using the HTTP REST API and performing authentication they will normally automatically grab the full REST API URL they are to use as provided by the authentication response. OpenStack's REST API allows for a client to authenticate to one URL and then be told to use a completely different URL for the actual service. Example: Client authenticates to https://identity.cloud.example.org:55443/v1/auth and gets a response with their authentication key and Storage URL (the URL of the proxy nodes or load balancer) of https://swift.cloud.example.org:44443/v1/AUTH_8980.

The method for configuring your web server to start and run as a non-root user varies by web server and OS.

負荷分散装置

If the option of using Apache is not feasible or for performance you wish to offload your SSL work you may employ a dedicated network device load balancer. This is also the common way to

provide redundancy and load balancing when using multiple proxy nodes.

If you choose to offload your SSL ensure that the network link between the load balancer and your proxy nodes is on a private (V)LAN segment such that other nodes on the network (possibly compromised) cannot wiretap (sniff) the unencrypted traffic. If such a breach were to occur the attacker could gain access to end-point client or cloud administrator credentials and access the cloud data.

The authentication service you use, such as Keystone or SWAuth, will determine how you configure a different URL in the responses to end-clients so they use your load balancer instead of an individual Proxy service node.

Object storage authentication

Object Storage uses wsgi to provide a middleware for authentication of end-point clients. The authentication provider defines what roles and user types exist. Some use traditional username and password credentials while others may leverage API key tokens or even client-side x.509 SSL certificates. Custom providers can be integrated in using the wsgi model.

Keystone

Keystone is the commonly used Identity provider in OpenStack. It may also be used for authentication in Object Storage. Coverage of securing Keystone is already provided in 18章Identity [77].

SWAuth

SWAuth is another alternative to Keystone. In contrast to Keystone it stores the user accounts, credentials, and metadata in object storage itself. More information can be found on the SWAuth website at http://gholt.github.io/swauth/.

他の重要事項

In /etc/swift/swift.conf on every service node there is a "swift_hash_path_suffix" setting. This is provided to reduce the

chance of hash collisions for objects being stored and avert one user overwriting the data of another user.

This value should be initially set with a cryptographically secure random number generator and consistent across all service nodes. Ensure that it is protected with proper ACLs and that you have a backup copy to avoid data loss.

第22章 ケーススタディ: ID 管理

このケーススタディでは、アリスとボブが OpenStack コアサービスの設定をどのように取り扱うかを議論します。これらには、Keystone Identity Service、Dashboard、Compute Services が含まれます。アリスは既存の政府ディレクトリサービスに統合することに関心があります。ボブはパブリックにアクセス権を提供する必要があります。

アリスのプライベートクラウド

アリスの企業はすべてのユーザーに対して 2 要素認証を持つディレクトリサービスが十分に確立されています。彼女は政府発行のアクセスカードを用いた認証をサポートする外部認証サービスをサポートするよう Keystone を設定します。アクセス制御ポリシーと統合されたユーザー用ロール情報を提供するために、外部 LDAP サービスも使用します。FedRAMP コンプライアンス要件のため、アリスはすべての管理アクセスに対して管理ネットワークで 2 要素認証を導入します。

アリスはクラウドのさまざまな観点を管理するために Dashboard も導入します。必ず HTTPS のみを使用するために HSTS と共に Dashboard を導入します。Dashboard はプライベートネットワークの DNS の内部サブドメインの中にあります。

アリスは仮想コンソールに VNC の代わりに SPICE を使用することを決めました。SPICE の先進的な機能の利点を得ようと思います。

ボブのパブリッククラウド

ボブは一般的なパブリックによる認証をサポートする必要があります。そのため、ユーザー名とパスワードによる認証を提供することを選択します。彼はユーザーのパスワードを解析しようとするブルートフォース攻撃について心配します。そのため、ログイン試行回数の失敗数を制限する外部認証拡張も使用します。ボブの管理ネットワークは彼のクラウドの中で他のネットワークと分離しています。しかし、彼の企業ネットワークから SSH 経由でアクセスできます。これまでに推奨しているとおり、ボブは管理者のパスワードが漏洩するリスクを減らすために、管理者が管理ネットワークで 2 要素認証を使用することを要求します。

ボブはクラウドのさまざまな観点を管理するために Dashboard も導入します。必ず HTTPS のみを使用するために HSTS と共に Dashboard を導

入します。Dashboard が同一オリジンポリシーの制限のため必ず第 2 レベルドメインに導入されるようにしました。また、リソース枯渇を防ぐために HORIZON_IMAGES_ALLOW_UPLOAD を無効化します。

ボブはその成熟度とセキュリティ機能から仮想コンソールに VNC を使用することを決めました。

第23章 ネットワークの状態

Grizzly リリースの OpenStack Networking により、エンドユーザーまたはテナントは、以前の OpenStack Networking リリースではできなかった新しい方法でネットワークリソースを定義、利用、消費することが可能です。OpenStack Networking は、ネットワーク設定のオーケストレーションに加えて、クラウド内のインスタンスを対象としたネットワーク接続の定義と IP アドレス指定用の対テナント API を提供します。API 中心のネットワークサービスへの移行にあたっては、クラウドのアーキテクトや管理者が、物理/仮想ネットワークのインフラストラクチャーとサービスをセキュリティ保護するためのベストプラクティスを考慮すべきです。

OpenStack Networking は、オープンソースコミュニティやサードパーティーのサービスによる API の拡張性を提供するプラグインアーキテクチャーで設計されました。アーキテクチャーの設計要件を評価するにあたっては、OpenStack Networking のコアサービスではどのような機能が提供されているか、サードパーティの製品によって提供される追加のサービスがあるかどうか、物理インフラストラクチャーにはどのような補足サービスを実装する必要があるかを判断することが重要です。

本項には、OpenStack Networking を実装する際に検討すべきプロセスとベストプラクティスについての大まかな概要をまとめています。提供されているサービスの現在の状況 、将来実装されるサービス、本プロジェクトにおける現在の制限事項などについて説明します。

第24章 Networking アーキテクチャ

OpenStack Networking は多数ノード間において幾つかのプロセスのデプロイにしばしば含まれる独立サービスです。OpenStack Networking サービスのメインプロセスは neutron-server で、これは OpenStack Networking API を提供し、追加処理用の適切なプラグインにテナントのリクエストを渡します。

OpenStack Networking コンポーネントは以下の要素を含みます。

- neutron サーバー (neutron-server と neutron-*-plugin): このサービスはネットワークノード上で実行され、Networking API とその拡張を提供します。これはまた、各ポートのネットワークモデルと IP アドレスを管理します。neutron-server とプラグインエージェントは、永続ストレージ用のデータベースへのアクセスと、内部通信用のメッセージキューへのアクセスを要求します。

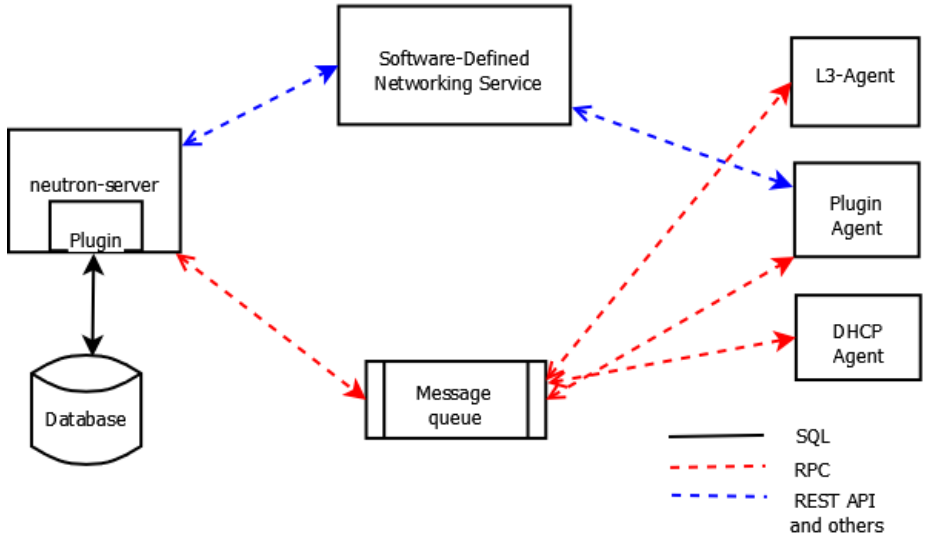
- プラグインエージェント (neutron-*-agent): ローカルの仮想スイッチ (vswitch) 設定を管理する為に各 compute ノード上で実行されます。実行するエージェントは、あなたが使用するプラグインに依存するでしょう。このサービスはメッセージキューへのアクセスを必要とします。オプションのプラグインに依存します。

- DHCP エージェント (neutron-dhcp-agent): テナントネットワークに DHCP サービスを提供します。このエージェントは全てのプラグインと同様で、DHCP 設定の管理を担当します。neutron-dhcp-agent はメッセージキューアクセスが必要です。

- L3 エージェント (neutron-l3-agent): テナントネットワーク上の VM において外部ネットワーク用 L3/NAT 転送を提供します。メッセージキューが必要です。プラグイン次第では別の物が必要になります。

- ネットワークプロバイダサービス (SDN サーバ／サービス)。テナントネットワークを提供する追加のネットワークサービスを提供します。これらの SDN サービスは REST API 又は他の通信チャネルを介して、neutron-server、neutron-plugin、プラグインエージェントと交信するかも知れません。

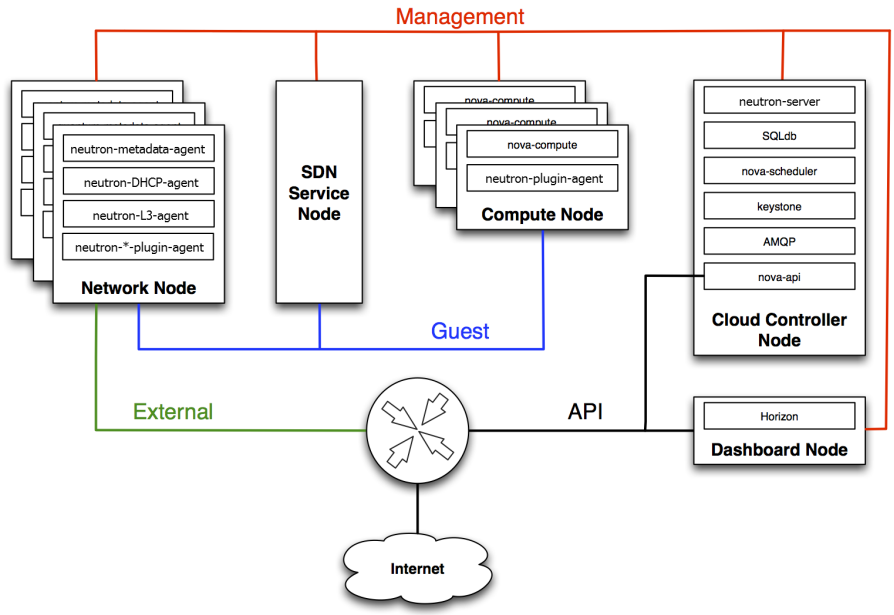
次表はOpenStack Networking コンポーネント群の構造・ネットワークフローダイアグラムを示しています。



OS ネットワーキングサービスの配置と物理サービス

このガイドでは、我々はまず、クラウドコントローラホスト１台、ネットワークホスト１台、VMを実行するcomputeハイパーバイザーの集合を含む標準的なアーキテクチャにフォーカスします。

物理サーバのネットワーク接続性



標準的な OpenStack Networking セットアップは最大４つの物理データセンターネットワークがあります。

- 管理ネットワーク OpenStack コンポーネント間の内部通信に使用されます。このネットワークの IP アドレスはデータセンター内でのみアクセス可能であるべきです。管理セキュリティドメインで検討します。

- ゲストネットワーク クラウドデプロイ中の VM データ通信に使用されます。このネットワークの IP アドレス要件は、使用中の OpenStack Networking プラグインとテナントにより作成される仮想ネットワークのネットワーク設定の選定に依存します。このネットワークはゲストセキュリティドメインで検討します。

- 外部ネットワーク 幾つかのデプロイシナリオ中のインターネットアクセスを持つVMを提供する為に使用されます。このネットワーク上の IP アドレスはインターネット上の誰もがアクセス可能です。パブリックセキュリティドメインで検討します。

- API ネットワーク テナントに OpenStack Networking API を含む全 OpenStack API を晒します。このネットワーク上の IP アドレスはインターネット上の誰もがアクセス可能であるべきです。これは外部

ネットワークと同じネットワークであっても構いません。外部ネットワーク用に、IP ブロック中の全 IP アドレス範囲より少ない部分を使う為の IP 割当範囲を使用するサブネットを作成する事が出来るからです。このネットワークはパブロックセキュリティドメインで検討します。

更なる情報は、OpenStack Cloud Administrator Guide 中の Networking の章を参照して下さい。

第25章 Networking サービス

あなたの OpenStack ネットワークインフラデザインの概要設計段階では、適切なセキュリティ管理・監査機構を確認する為、物理ネットワークインフラ設計で支援する適切な専門技術が間違いなく利用できる事は重要です。

OpenStack Networking は（テナントに自身の仮想ネットワークを設計する為の機能を提供する）仮想ネットワークサービスのレイヤを追加します。これらの仮想化サービスは、現時点で従来のネットワークコンポーネントのように成熟していません。これらの仮想化技術の現状と、仮想ネットワークと従来のネットワーク境界でどのコントロールを実装する必要があるだろうというを知っておく事は重要です。

VLAN とトンネリングを使用した L2 分断

OpenStack Networking はテナント／ネットワークの組合せ単位で通信を分断する為の、 VLANs（IEEE 802.1Q タギング）又は GRE カプセル化を使用した L2 トンネルという２つの異なる機構を使用する事が出来ます。通信の分断と独立用にあなたが選択する方式は、あなたの OpenStack デプロイの範囲と規模に依存します。

VLAN

VLAN は特別な VLAN ID（VID）フィールド値を持つ IEEE 802.1Q ヘッダを含む特別な物理ネットワーク上のパケットを実現します。同じ物理ネットワークを共有する VLAN ネットワーク群は、L2 において相互から独立しており、重複する IP アドレス空間を持つ事すら可能です。VLAN ネットワークに対応した各個別の物理ネットワークは、独自の VID 値を持つ独立した VLAN トランクとして扱われます。有効な VID 値は1～4094です。

VLAN 設定の複雑さはあなたの OpenStack 設計要件に依存します。OpenStack Networking がVLAN を効率良く使用できるようにする為に、VLAN 範囲を（各テナントに１つ）割り当てて、各 compute ノードの物理スイッチポートを VLAN トランクポートに変更する必要があります。

注記

注意：あなたのネットワークを4095 以上のテナントに対応するようにしたい場合、VLAN はあなたにとって多分正しい選択肢ではありません。なぜなら、4095 以上に VLAN タグを拡張する為の複数の「改造」が必要だからです。

L2 トンネリング

Network tunneling encapsulates each tenant/network combination with a unique "tunnel-id" ネットワークトンネリングは、固有の「トンネルID」を用いてテナント／ネットワークの各組合せをカプセル化します。これは、上記の組合せに属するネットワーク通信を独立させる為に使用されます。テナントの L2 ネットワーク接続は、物理的配置や下層のネットワーク設計から独立しています。IP パケット内で通信をカプセル化する事により、通信はレイヤ３境界を越える事ができ、VLAN や VLAN とランキングの事前設定の必要が無くなります。トンネリングはネットワークのデータ通信に不明瞭なレイヤを追加し、監視の観点で個々のテナント通信の可視性を低下させます。

OpenStack Networking は現在 GRE カプセル化のみサポートしており、Havana リリースで VXLAN をサポートする計画があります。

L2 分断を提供する技術の選択は、あなたのデプロイで作成される予定のテナントネットワークの範囲とサイズに依存します。あなたの環境が VLAN ID の利用で制限がある場合や、大多数の L2 ネットワークが見込まれる場合、トンネリングの使用を推奨します。

ネットワークサービス

テナントネットワーク分断の選択はネットワークセキュリティと制御境界をどのように実装するかに影響します。 以下の追加ネットワークサービスは利用可能か、OpenStack ネットワークアーキテクチャのセキュリティポーズを拡張する為の開発中かのいずれかです。

アクセスコントロールリスト

OpenStack Compute は、旧式の nova-network サービスでデプロイする場合、テナントネットワーク通信のアクセス制御を直接サポートします。又は、OpenStack Networking サービスにアクセス制御を任せる事も出来ます。

注：旧式の nova-network セキュリティグループは、Iptables を使用してインスタンス上の全ての仮想インターフェースポートに適用されます。

セキュリティグループでは、管理者とテナントが仮想インターフェースポート通過を許可する通信のタイプと方向（内向き／外向き）を指定できるようになっています。

OpenStack Networking 経由でセキュリティグループを有効にする事をお勧めします。

L3 ルーティングおよび NAT

OpenStack Networking のルータは複数の L2 ネットワークを接続でき、１つ以上のプライベート L2 ネットワークを共有外部ネットワーク（インターネットアクセス用のパブリックネットワーク等）に接続するゲートウェイを提供する事も出来ます。

L3 ルータは、外部ネットワークへのルータに接続するゲートウェイポート上の基本的なネットワークアドレス変換（NAT）機能を提供します。このルータはデフォルトで全てのネットワークの SNAT（静的 NAT）を行います。これは、外部ネットワーク上のパブリック IP アドレスから、ルータにアタッチされた他の１サブネットのプライベート IP アドレスへ変換する静的な１対１マッピングを作成します。

テナント VM のより粒度の細かいテナント L3 ルーティングとフローティング IP 単位で設定する事をお勧めします。

サービス品質(QoS)

現在の OpenStack Networking にはテナントインスタンスの仮想インターフェースポート上の QoS 設定機能が欠如しています。物理ネットワークエッジデバイスにおけるトラフィックシェーピングやレートリミットの為の QoS 活用は、OpenStack デプロイ中のワークロードの動的な性質の為に実装されておらず、従来の方法では設定できません。QoS-as-a-Service（QoSaaS）は実験的な機能として現在 OpenStack Networking Havana リリース用に開発中です。QoSaaS は以下のサービスを提供する計画です。

- DSCP マーキングによるトラフィックシェーピング
- ポート・ネットワーク・テナント単位のレートリミット
- ポートミラーリング（オープンソースのサードパーティ製プラグイン使用）

- フロー分析（オープンソースのサードパーティプラグイン使用）

テナントトラフィックポートミラーリング又はNetwork Flow モニタリングは現在、OpenStack Networking の機能として公開されていません。ポート／ネットワーク／テナント単位でポートミラーリングを行うサードパーティ製のプラグイン拡張があります。ハイパーバイザー上で Open vSwitch を使用する場合、sFlow とポートミラーリングを有効にできますが、実装には幾つかの運用操作が必要になるでしょう。

ロードバランシング

OpenStack Networking の Grizzly リリースにおける実験的機能の１つが Load-Balancer-as-a-service（LBaaS）です。LBaaS API は、アーリーアダプターやベンダーに LBaaS 技術の実装を行う機会を提供します。しかしながら、リファレンス実装は未だ実験段階で、商用環境で使用されているという話は聞きません。現在のリファレンス実装は HAProxy をベースにしています。仮想インターフェースポート用の拡張可能な L4-L7 機能を提供する OpenStack Networking 中の拡張用に開発中のサードパーティプラグインがあります。

ファイアウォール

FW-as-a-Service（FWaaS）は実験的機能として OpenStack Networking Havana リリースに向けて現在開発中です。FWaaS は現在セキュリティグループにより提供されるものより一般にはかなり広い典型的なファイアウォール製品により提供される豊富なセキュリティ機能を管理・設定する為に呼ばれます。現在、FWaaS をサポートするために、OpenStack ネットワーキングの拡張用サードパーティプラグインが開発されているところです。

利用可能なネットワークサービスの現在の機能と制限を理解する事は OpenStack Networking の設計上極めて重要です。仮想／物理ネットワークの境界がどこかを理解する事は、あなたの環境で要求されたセキュリティコントロールを追加する際の助けになるでしょう。

ネットワークサービス拡張

以下はオープンソースコミュニティ又はSDN企業によって提供された、OpenStack Networking で動作する既知のプラグインの一覧です。

Big Switch Controller Plugin, Brocade Neutron Plugin Brocade Neutron Plugin, Cisco UCS/Nexus Plugin, Cloudbase Hyper-V Plugin, Extreme Networks Plugin, Juniper Networks Neutron Plugin, Linux

Bridge Plugin, Mellanox Neutron Plugin, MidoNet Plugin, NEC OpenFlow Plugin, Open vSwitch Plugin, PLUMgrid Plugin, Ruijie Networks Plugin, Ryu OpenFlow Controller Plugin, VMware NSX plugin

For a more detailed comparison of all features provided by plugins as of the Folsom release, see Sebastien Han's comparison.

Networking サービスの制限事項

OpenStack Networking は以下の制限があります。

- Overlapping IP addresses — If nodes that run either neutron-l3-agent or neutron-dhcp-agent use overlapping IP addresses, those nodes must use Linux network namespaces. By default, the DHCP and L3 agents use Linux network namespaces. However, if the host does not support these namespaces, run the DHCP and L3 agents on different hosts.

  If network namespace support is not present, a further limitation of the L3 Agent is that only a single logical router is supported.

- Multi-Host DHCP-agent — OpenStack Networking supports multiple l3-agent and dhcp-agents with load balancing. However, tight coupling of the location of the virtual machine is not supported.

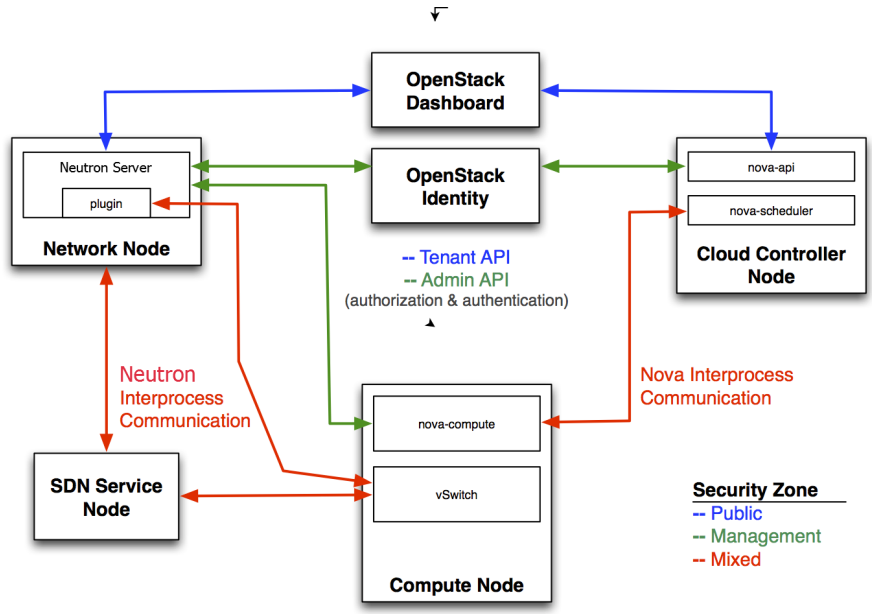
- No IPv6 Support for L3 agents — The neutron-l3-agent, used by many plugins to implement L3 forwarding, supports only IPv4 forwarding.

第26章 Securing OpenStack Networking Services

In order to secure OpenStack Networking, an understanding of the workflow process for tenant instance creation needs to be mapped to security domains.

There are four main services that interact with OpenStack Networking. In a typical OpenStack deployment these services map to the following security domains:

- OpenStack Dashboard: Public and Management
- OpenStack Identity: Management
- OpenStack Compute Node: Management and Guest
- OpenStack Network Node: Management, Guest, and possibly Public depending upon neutron-plugin in use.
- SDN Services Node: Management, Guest and possibly Public depending upon product used.



In order to isolate sensitive data communication between the OpenStack Networking services and other OpenStack core services, we strongly recommend that these communication channels be configured to only allow communications over an isolated management network.

OpenStack Networking Service Configuration

Restrict Bind Address of the API server: neutron-server

To restrict the interface or IP address on which the OpenStack Networking API service binds a network socket for incoming client connections, specify the bind_host and bind_port in the neutron.conf file as shown:

```
# Address to bind the API server
bind_host = <ip address of server>

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

Restrict DB and RPC communication of the OpenStack Networking services:

Various components of the OpenStack Networking services use either the messaging queue or database connections to communicate with other components in OpenStack Networking.

It is recommended that you follow the guidelines provided in the Database Authentication and Access Control chapter in the Database section for all components that require direct DB connections.

It is recommended that you follow the guidelines provided in the Queue Authentication and Access Control chapter in the Messaging section for all components that require RPC communication.

第27章 Networking Services Security Best Practices

This section discusses OpenStack Networking configuration best practices as they apply to tenant network security within your OpenStack deployment.

Tenant Network Services Workflow

OpenStack Networking provides users real self services of network resources and configurations. It is important that Cloud Architects and Operators evaluate their design use cases in providing users the ability to create, update, and destroy available network resources.

Networking Resource Policy Engine

A policy engine and its configuration file, policy.json, within OpenStack Networking provides a method to provide finer grained authorization of users on tenant networking methods and objects. It is important that cloud architects and operators evaluate their design and use cases in providing users and tenants the ability to create, update, and destroy available network resources as it has a tangible effect on tenant network availability, network security, and overall OpenStack security. For a more detailed explanation of OpenStack Networking policy definition, please refer to the Authentication and authorization section in the OpenStack Cloud Administrator Guide.

It is important to review the default networking resource policy and modify

If your deployment of OpenStack provides multiple external access points into different security domains it is important that you limit the tenant's ability to attach multiple vNICs to multiple

external access points -- this would bridge these security domains and could lead to unforeseen security compromise. It is possible mitigate this risk by utilizing the host aggregates functionality provided by OpenStack Compute or through splitting the tenant VMs into multiple tenant projects with different virtual network configurations.

セキュリティグループ

The OpenStack Networking Service provides security group functionality using a mechanism that is more flexible and powerful than the security group capabilities built into OpenStack Compute. Thus, when using OpenStack Networking, nova.conf should always disable built-in security groups and proxy all security group calls to the OpenStack Networking API. Failure to do so will result in conflicting security policies being simultaneously applied by both services. To proxy security groups to OpenStack Networking, use the following configuration values:

- firewall_driver : must be set to 'nova.virt.firewall.NoopFirewallDriver' so that nova-compute does not perform iptables-based filtering itself.
- security_group_api : must be set to 'neutron' so that all security group requests are proxied to the OpenStack Network Service.

Security groups and security group rules allow administrators and tenants the ability to specify the type of traffic and direction (ingress/egress) that is allowed to pass through a virtual interface port. A security group is a container for security group rules. When a virtual interface port is created in OpenStack Networking it is associated with a security group. If a security group is not specified, the port will be associated with a 'default' security group. By default this group will drop all ingress traffic and allow all egress. Rules can be added to this group in order to change the behaviour.

When using the security group API through OpenStack Compute, security groups are applied to all virtual interface ports on an instance. The reason for this is that OpenStack Compute security group APIs are instance based and not virtual interface port based as OpenStack Networking.

クォータ

Quotas provide the ability to limit the number of network resources available to tenants. You can enforce default quotas for all tenants.

```
/etc/neutron/neutron.conf
[QUOTAS]
# resource name(s) that are supported in quota features
quota_items = network,subnet,port

# default number of resource allowed per tenant, minus for unlimited
#default_quota = -1

# number of networks allowed per tenant, and minus means unlimited
quota_network = 10

# number of subnets allowed per tenant, and minus means unlimited
quota_subnet = 10

# number of ports allowed per tenant, and minus means unlimited
quota_port = 50

# number of security groups allowed per tenant, and minus means unlimited
quota_security_group = 10

# number of security group rules allowed per tenant, and minus means
 unlimited
quota_security_group_rule = 100

# default driver to use for quota checks
quota_driver = neutron.quota.ConfDriver
```

OpenStack Networking also supports per-tenant quotas limit via a quota extension API. To enable per-tenant quotas, you need to set quota_driver in neutron.conf.

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

第28章 Case Studies: Networking

In this case study we discuss how Alice and Bob would address providing networking services to the user.

アリスのプライベートクラウド

A key objective of Alice's cloud is to integrate with the existing auth services and security resources. The key design parameters for this private cloud are a limited scope of tenants, networks and workload type. This environment can be designed to limit what available network resources are available to the tenant and what are the various default quotas and security policies are available. The network policy engine can be modified to restrict creation and changes to network resources. In this environment, Alice might want to leverage nova-network in the application of security group polices on a per instance basis vs. Neutron's application of security group polices on a per port basis. L2 isolation in this environment would leverage VLAN tagging. The use of VLAN tags will allow great visibility of tenant traffic by leveraging existing features and tools of the physical infrastructure.

ボブのパブリッククラウド

A major business driver for Bob is to provide an advanced networking services to his customers. Bob's customers would like to deploy multi-tiered application stacks. This multi-tiered application are either existing enterprise application or newly deployed applications. Since Bob's public cloud is a multi-tenancy enterprise service, the choice to use for L2 isolation in this environment is to use overlay networking. Another aspect of Bob's cloud is the self-service aspect where the customer can provision available networking services as needed. These networking services encompass L2 networks, L3 Routing, Network ACL and NAT. It is important that per-tenant quota's be implemented in this environment.

An added benefit with utilizing OpenStack Networking is when new advanced networking services become available, these new features can be easily provided to the end customers.

第29章 メッセージキューアーキテクチャー

Message queuing services facilitate inter-process communication in OpenStack. OpenStack supports these message queuing service back ends:

- RabbitMQ
- Qpid
- ZeroMQ or 0MQ

Both RabbitMQ and Qpid are Advanced Message Queuing Protocol (AMQP) frameworks, which provide message queues for peer-to-peer communication. Queue implementations are typically deployed as a centralized or decentralized pool of queue servers. ZeroMQ provides direct peer-to-peer communication through TCP sockets.

Message queues effectively facilitate command and control functions across OpenStack deployments. Once access to the queue is permitted no further authorization checks are performed. Services accessible through the queue do validate the contexts and tokens within the actual message payload. However, you must note the expiration date of the token because tokens are potentially re-playable and can authorize other services in the infrastructure.

OpenStack does not support message-level confidence, such as message signing. Consequently, you must secure and authenticate the message transport itself. For high-availability (HA) configurations, you must perform queue-to-queue authentication and encryption.

With ZeroMQ messaging, IPC sockets are used on individual machines. Because these sockets are vulnerable to attack, ensure that the cloud operator has secured them.

第30章 メッセージングのセキュリティ

この章では、OpenStack で使用される最も一般的なメッセージキュー製品である、Rabbit MQ、Qpid、ZeroMQ の堅牢化アプローチについて説明します。

メッセージ通信路のセキュリティ

AMQP ベースの製品（Qpid, RabbitMQ）は SSL を用いた通信路レベルのセキュリティに対応しています。ZeroMQ はSSL をネイティブでサポートしていませんが、Labeled-IPSec や CIPSO ネットワークラベルを用いた通信路レベルのセキュア化に対応しています。

メッセージキューには、通信路レベルでの暗号化を強く推奨します。メッセージクライアントとの接続に SSL を用いることで、メッセージサーバとの通信路における通信の改ざんや傍受を防ぐことが可能です。以下、よく使われる 2 種類のメッセージサーバ Qpid、および、RabbitMQ における一般的な SSL の設定について説明します。 クライアント接続の正当性を保証する目的でメッセージサーバに証明機関 (CA) バンドルを設定する場合、該当ノードに限定した CA の使用を、またなるべくなら組織内部で管理している CA の使用を推奨します。 信頼された CA バンドルは許可を与えるクライアント接続証明書を決定し、SSL 接続を張るためのクライアントサーバ検証のステップを通過させます。 証明書とキーのファイルをインストールする際は、chmod 0600 などでファイルのパーミッションを限定させ、所有者をメッセージサーバのデーモンユーザに限定させるようにしてください。こうすることで、メッセージサーバ上の許可を与えていない他プロセスやユーザによるアクセスを防ぐことできます。

RabbitMQ サーバ SSL 設定

下記の設定を RabbitMQ のシステム設定ファイルに追加します。通常、/etc/rabbitmq/rabbitmq.conf に保存されています。

```
[
```

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

'tcp_listeners' オプションを '[]' に指定し、非 SSL ポートの接続を受け付けない設定にしていることに注意してください。'ssl_listeners' オプションはサービスの管理ネットワークのみ受け付けるよう限定すべきです。

RabbitMQ の SSL 設定に関する詳細は、以下を参照してください。

- RabbitMQ 設定
- RabbitMQ SSL

Qpid サーバ SSL 設定

Apache Foundation が Qpid のメッセージングセキュリティガイドを発行しています。

- Apache Qpid SSL

キューの認証およびアクセス制御

RabbitMQ と Qpid はキューへのアクセス制御を目的とした、認証およびアクセス制御の仕組みを持っています。ZeroMQ にはこのような仕組みは備わっていません。

Simple Authentication and Security Layer (SASL) はインターネットプロトコルにおける認証とデータセキュリティのフレームワークです。RabbitMQ と Qpid は SASL の他、プラグイン形式の認証メカニズムを提供しており、単純なユーザ名とパスワードよりもセキュアな認証が可能になっています。RabbitMQ は SASL をサポートしているものの、現在の OpenStack は特定の SASL 認証メカニズムの使用を許可していません。RabbitMQ では、非暗号化接続でのユーザ名とパスワード認証か、X.509 クライアント証明書を用いたセキュアな SSL 接続でのユーザ名とパスワード認証がサポートされています。

全ての OpenStack サービスノードにおいて、メッセージキューへのクライアント接続に X.509 クライアント証明書を設定することを推奨します。また可能なら、X.509 クライアント証明書での認証も推奨します。(現在、Qpid のみがサポート) ユーザ名とパスワードを用いる場合、キューに対するアクセスの監査の粒度を細かくする目的で、アカウントはサービス毎、ノード毎に作成するべきです。

また、キューサーバが使用する SSL ライブラリについても展開の前に考慮しておく必要があります。Qpid はMozilla の NSS ライブラリを、RabbitMQ は OpenSSL を使う Erlang の SSL モジュールを用いています。

認証設定例 - RabbitMQ

RabbitMQ サーバで、デフォルトの 'guest' ユーザを削除します。

```
rabbitmqctl delete_user quest
```

RabbitMQ サーバにて、メッセージキューを使用する各 OpenStack サービス、または、ノード毎にユーザアカウントと権限を設定します。

```
rabbitmqctl add_user compute01 password
rabbitmqctl set_permissions compute01 ".*"".*"".*"
```

追加の設定情報は以下を参照してください。

- RabbitMQ アクセス制御
- RabbitMQ 認証
- RabbitMQ プラグイン
- RabbitMQ SASL 外部認証

OpenStack サービス設定 - RabbitMQ

```
[DEFAULT]
rpc_backend=nova.openstack.common.rpc.impl_kombu
rabbit_use_ssl=True
rabbit_host=
rabbit_port=5671
rabbit_user=compute01
rabbit_password=password
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 : Grizzly バージョンでは、設定ファイルに "kombu_ssl_version" が定義されていると、下記の Python トレースバックエラーが発生します。 'TypeError: an integer is required' "kombu_ssl_version" を設定ファイルから削除することで、このエラーを防ぐことが可能です。現在の状況は、bug report 1195431 https://bugs.launchpad.net/oslo/+bug/1195431 を参照してください。

認証設定例 - Qpid

設定情報は以下を参照してください。

- Apache Qpid 認証
- Apache Qpid 認可

OpenStack サービス設定 - Qpid

```
[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=5671qpid_username=compute01

qpid_password=password
```

オプションとして Qpid で SASL を使用する場合は、下記のように SASL メカニズムを指定します。

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

メッセージキュープロセスのアイソレーションとポリシー

各プロジェクトは多数のサービスを提供し、それぞれがメッセージを送信、消費します。メッセージを送信した各バイナリは、リプライのみの場合、該当キューからメッセージを消費するはずです。

メッセージキューサービスのプロセスは、他のキューサービスのプロセスや、同一マシン上の他プロセスと分離すべきです。

名前空間

ネットワーク名前空間の設定は、OpenStack コンピュートハイパーバイザを動作させる全てのサービスで強く推奨します。ネットワーク名前空間を用いることで、VM ゲストと管理ネットワークのトラフィックがブリッジングされることを防ぎます。

ZeroMQ メッセージングを使用する場合、ネットワーク経由のメッセージ受信と、IPC経由によるローカルプロセスへのメッセージ送信のために、各ホストに最低 1 つの ZeroMQ メッセージレシーバーを走らせる必要があります。IPC 名前空間内にプロジェクト毎で独立したメッセージレシーバーを構築することが可能であり望ましいです。また同様に、同一プロジェクト内でも異なるサービスごとに独立したメッセージレシーバーを構築することが望ましいです。

ネットワークポリシー

キューサーバーは管理ネットワークからの接続のみを受け付けるべきであり、この方針はあらゆる実装に適用されます。サービスの設定を通して実装し、任意でグローバルネットワークポリシーを追加で実装します。

ZeroMQ を使用するのであれば、各プロジェクトで独立した専用のポート上で動作する ZeroMQ レシーバープロセスを用意すべきです。これは、AMQP のコントロール exchange の概念に相当します。

強制アクセス制御

各プロセスに行なった設定は、他プロセスに影響を与えないよう制限をかけるべきです。そのためには、ダイレクトアクセス制御のみではなく、強制アクセス制御を使用します。このような制限をかけるのは、同一マシンで動作する他プロセスとの隔離を防ぐことが目的です。

第31章 ケーススタディ: メッセージング

メッセージキューは、多数の OpenStack サービスを支える重要なインフラストラクチャであり、特にコンピュートサービスと強く結びついています。メッセージキューサービスの性質上、アリスとボブが抱えるセキュリティ上の懸念はよく似ています。特に大きな残課題は、数多くのシステムがキューにアクセスしているものの、キューメッセージのコンシューマーには、キューを発行したホストやサービスを確かめる手立てがないことです。攻撃者がキューの発行に成功すると、仮想マシンの作成や削除をしたり、あらゆるテナントのブロックストレージに接続するなど、他にも無数の悪意のある攻撃が可能になってしまいます。 これを防ぐためのソリューションが出始めており、いくつかはメッセージへの署名と暗号化を使ったものが OpenStack の開発プロセスで進んでいます。

アリスのプライベートクラウド

このケースでは、アリスの方法はボブがパブリッククラウドに展開した方法と同じものを使用します。

ボブのパブリッククラウド

ボブは、コンピュートサービスを支えるインフラストラクチャとネットワークがある時点でセキュリティ侵害に会うと仮定します。そして、メッセージキューへのアクセス制限の重要性に気づきました。 そこで、RabbitMQ サーバーに SSL と X.509 クライアントアクセス制御を適用することにします。これにより、キューアクセスを持たないシステムを乗っ取られても、攻撃者の能力を制限することができます。

さらにボブは、メッセージサーバーと通信できるエンドポイントを、強力なネットワークの ACL ルールセットで制限することにしました。この2個目の制限が、他の防御が失敗した場合の保険として機能します。

第32章 Database Backend Considerations

The choice of database server is an important consideration in the security of an OpenStack deployment. While security considerations are not the only basis on which a database server must be chosen, security considerations are the only ones within the scope of this book. In practice, OpenStack only supports two database types: PostgreSQL and MySQL.

PostgreSQL has a number of desirable security features such as Kerberos authentication, object-level security, and encryption support. The PostgreSQL community has done well to provide solid guidance, documentation, and tooling to promote positive security practices.

MySQL has a large community, wide-spread adoption, and provides high availability options. MySQL also has the ability to provide enhanced client authentication by way of plug-in authentication mechanisms. Forked distributions in the MySQL community provide many options for consideration. It is important to choose a specific implementation of MySQL based on a thorough evaluation of the security posture and the level of support provided for the given distribution.

Security References for Database Backends

Those deploying MySQL or PostgreSQL are advised to refer to existing security guidance. Some references are listed below:

MySQL:

- OWASP MySQL Hardening
- MySQL Pluggable Authentication
- Security in MySQL

PostgreSQL:

- OWASP PostgreSQL Hardening
- Total security in a PostgreSQL database

第33章 Database Access Control

Each of the core OpenStack services (Compute, Identity, Networking, Block Storage) store state and configuration information in databases. In this chapter, we discuss how databases are used currently in OpenStack. We also explore security concerns, and the security ramifications of database backend choices.

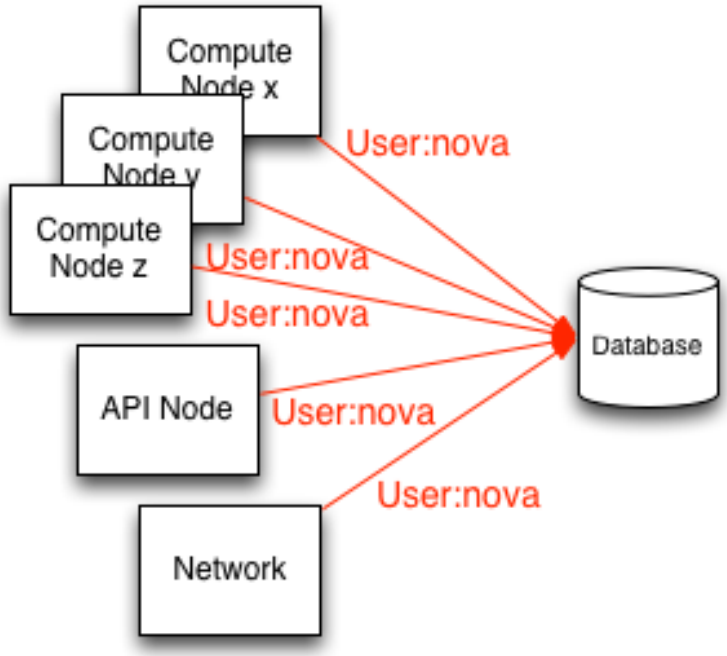
OpenStack Database Access Model

All of the services within an OpenStack project access a single database. There are presently no reference policies for creating table or row based access restrictions to the database.

There are no general provisions for granular control of database operations in OpenStack. Access and privileges are granted simply based on whether a node has access to the database or not. In this scenario, nodes with access to the database may have full privileges to DROP, INSERT, or UPDATE functions.

Granular Access Control

By default, each of the OpenStack services and their processes access the database using a shared set of credentials. This makes auditing database operations and revoking access privileges from a service and its processes to the database particularly difficult.

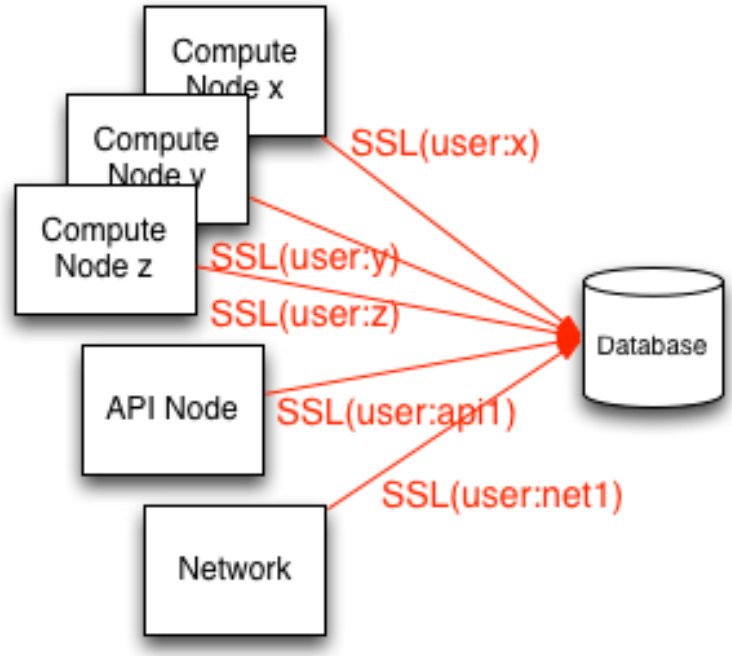


Nova Conductor

The compute nodes are the least trusted of the services in OpenStack because they host tenant instances. The nova-conductor service has been introduced to serve as a database proxy, acting as an intermediary between the compute nodes and the database. We discuss its ramifications later in this chapter.

We strongly recommend:

- All database communications be isolated to a management network
- Securing communications using SSL
- Creating unique database user accounts per OpenStack service endpoint (illustrated below)



Database Authentication and Access Control

Given the risks around access to the database, we strongly recommend that unique database user accounts be created per node needing access to the database. Doing this facilitates better analysis and auditing for ensuring compliance or in the event of a compromise of a node allows you to isolate the compromised host by removing access for that node to the database upon detection. When creating these per service endpoint database user accounts, care should be taken to ensure that they are configured to require SSL. Alternatively, for increased security it is recommended that the database accounts be configured using X.509 certificate authentication in addition to usernames and passwords.

Privileges

A separate database administrator (DBA) account should be created and protected that has full privileges to create/drop databases, create user accounts, and update user privileges. This simple means of separation of responsibility helps prevent accidental misconfiguration, lowers risk and lowers scope of compromise.

The database user accounts created for the OpenStack services and for each node should have privileges limited to just the database relevant to the service where the node is a member.

Require User Accounts to Require SSL Transport

Configuration Example #1: (MySQL)

```
GRANT ALL ON dbname.* to 'compute01'@'hostname' IDENTIFIED BY 'password'
 REQUIRE SSL;
```

Configuration Example #2: (PostgreSQL)

In file pg_hba.conf:

```
hostssl dbname compute01 hostname md5
```

Note this command only adds the ability to communicate over SSL and is non-exclusive. Other access methods that may allow unencrypted transport should be disabled so that SSL is the sole access method.

The 'md5' parameter defines the authentication method as a hashed password. We provide a secure authentication example in the section below.

Authentication with X.509 Certificates

Security may be enhanced by requiring X.509 client certificates for authentication. Authenticating to the database in this

manner provides greater identity assurance of the client making the connection to the database and ensures that the communications are encrypted.

Configuration Example #1: (MySQL)

```
GRANT ALL on dbname.* to 'compute01'@'hostname' IDENTIFIED BY 'password'
 REQUIRE SUBJECT
'/C=XX/ST=YYY/L=ZZZZ/O=cloudycloud/CN=compute01' AND ISSUER
'/C=XX/ST=YYY/L=ZZZZ/O=cloudycloud/CN=cloud-ca';
```

Configuration Example #2: (PostgreSQL)

```
hostssl dbname compute01 hostname cert
```

OpenStack Service Database Configuration

If your database server is configured to require X.509 certificates for authentication you will need to specify the appropriate SQLAlchemy query parameters for the database backend. These parameters specify the certificate, private key, and certificate authority information for use with the initial connection string.

Example of an :sql_connection string for X.509 certificate authentication to MySQL:

```
sql_connection = mysql://compute01:password@localhost/nova?
charset=utf8&ssl_ca=/etc/mysql/cacert.pem&ssl_cert=/etc/mysql/server-cert.
pem&ssl_key=/etc/mysql/server-key.pem
```

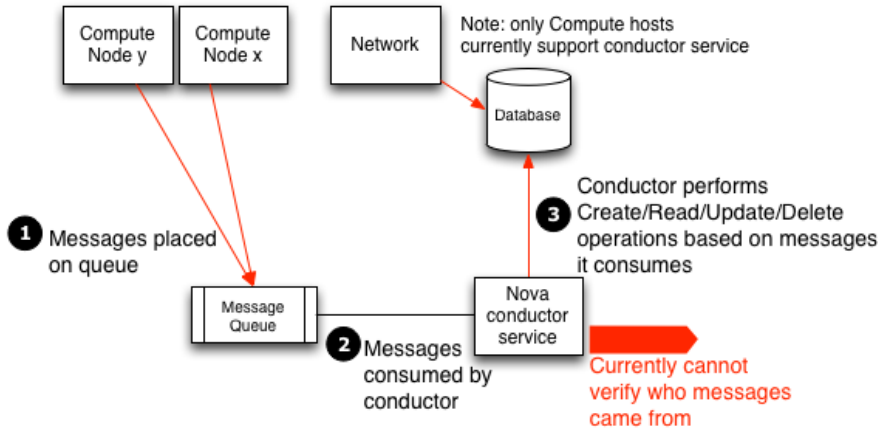
Nova Conductor

OpenStack Compute offers a sub-service called nova-conductor which proxies database connections, with the primary purpose of having the nova compute nodes interfacing with nova-conductor to meet data persistence needs as opposed to directly communicating with the database.

Nova-conductor receives requests over RPC and performs actions on behalf of the calling service without granting granular access

to the database, its tables, or data within. Nova-conductor essentially abstracts direct database access away from compute nodes.

This abstraction offers the advantage of restricting services to executing methods with parameters, similar to stored procedures, preventing a large number of systems from directly accessing or modifying database data. This is accomplished without having these procedures stored or executed within the context or scope of the database itself, a frequent criticism of typical stored procedures.



Unfortunately, this solution complicates the task of more fine-grained access control and the ability to audit data access. Because the nova-conductor service receives requests over RPC, it highlights the importance of improving the security of messaging. Any node with access to the message queue may execute these methods provided by the nova-conductor and effectively modifying the database.

Finally, it should be noted that as of the Grizzly release, gaps exist where nova-conductor is not used throughout OpenStack Compute. Depending on one's configuration, the use of nova-conductor may not allow deployers to avoid the necessity of providing database GRANTs to individual compute host systems.

Note, as nova-conductor only applies to OpenStack Compute, direct database access from compute hosts may still be necessary for the operation of other OpenStack components such as Telemetry (Ceilometer), Networking, and Block Storage.

Implementors should weigh the benefits and risks of both configurations before enabling or disabling the nova-conductor service. We are not yet prepared to recommend the use of nova-conductor in the Grizzly release. However, we do believe that this recommendation will change as additional features are added into OpenStack.

To disable the nova-conductor, place the following into your nova.conf file (on your compute hosts):

```
[conductor]
use_local = true
```

第34章 Database Transport Security

This chapter covers issues related to network communications to and from the database server. This includes IP address bindings and encrypting network traffic with SSL.

Database Server IP Address Binding

To isolate sensitive database communications between the services and the database, we strongly recommend that the database server(s) be configured to only allow communications to and from the database over an isolated management network. This is achieved by restricting the interface or IP address on which the database server binds a network socket for incoming client connections.

Restricting Bind Address for MySQL

In my.cnf:

```
[mysqld]
...
bind-address <ip address or hostname of management network interface>
```

Restricting Listen Address for PostgreSQL

In postgresql.conf:

```
listen_addresses = <ip address or hostname of management network interface>
```

Database Transport

In addition to restricting database communications to the management network, we also strongly recommend that the cloud

administrator configure their database backend to require SSL. Using SSL for the database client connections protects the communications from tampering and eavesdropping. As will be discussed in the next section, using SSL also provides the framework for doing database user authentication via X.509 certificates (commonly referred to as PKI). Below is guidance on how SSL is typically configured for the two popular database backends MySQL and PostgreSQL.

注記

NOTE: When installing the certificate and key files, ensure that the file permissions are restricted, for example chmod 0600, and the ownership is restricted to the database daemon user to prevent unauthorized access by other processes and users on the database server.

MySQL SSL Configuration

The following lines should be added in the system-wide MySQL configuration file:

In my.cnf:

```
[[mysqld]]
...
ssl-ca=/path/to/ssl/cacert.pem
ssl-cert=/path/to/ssl/server-cert.pem
ssl-key=/path/to/ssl/server-key.pem
```

Optionally, if you wish to restrict the set of SSL ciphers used for the encrypted connection. See http://www.openssl.org/docs/apps/ciphers.html for a list of ciphers and the syntax for specifying the cipher string:

```
ssl-cipher='cipher:list'
```

PostgreSQL SSL Configuration

The following lines should be added in the system-wide PostgreSQL configuration file, postgresql.conf.

```
ssl = true
```

Optionally, if you wish to restrict the set of SSL ciphers used for the encrypted connection. See http://www.openssl.org/docs/apps/ciphers.html for a list of ciphers and the syntax for specifying the cipher string:

```
ssl-ciphers = 'cipher:list'
```

The server certificate, key, and certificate authority (CA) files should be placed in the $PGDATA directory in the following files:

- $PGDATA/server.crt - Server certificate
- $PGDATA/server.key - Private key corresponding to server.crt
- $PGDATA/root.crt - Trusted certificate authorities
- $PGDATA/root.crl - Certificate revocation list

第35章 Case Studies: Database

In this case study we discuss how Alice and Bob would address database selection and configuration for their respective private and public clouds.

アリスのプライベートクラウド

Alice's organization has high availability concerns, so she has elected to use MySQL for the database. She further places the database on the Management network and uses SSL with mutual authentication among the services to ensure secure access. Given there will be no external access of the database, she uses certificates signed with the organization's self-signed root certificate on the database and its access endpoints. Alice creates separate user accounts for each database user, and configures the database to use both passwords and X.509 certificates for authentication. She elects not to use the nova-conductor sub-service due to the desire for fine-grained access control policies and audit support.

ボブのパブリッククラウド

Bob is concerned about strong separation of his tenants' data, so he has elected to use the Postgres database , known for its stronger security features. The database resides on the Management network and uses SSL with mutual authentication with the services. Since the database is on the Management network, the database uses certificates signed with the company's self-signed root certificate. Bob creates separate user accounts for each database user, and configures the database to use both passwords and X.509 certificates for authentication. He elects not to use the nova-conductor sub-service due to a desire for fine-grained access control.

第36章 Data Privacy Concerns

OpenStack is designed to support multitenancy and those tenants will most probably have different data requirements. As a cloud builder and operator you need to ensure your OpenStack environment can address various data privacy concerns and regulations. In this chapter we will address the following topics around Data Privacy as it pertains to OpenStack implementations:

- Data Residency
- Data Disposal

Data Residency

The privacy and isolation of data has consistently been cited as the primary barrier to cloud adoption over the past few years. Concerns over who owns data in the cloud and whether the cloud operator can be ultimately trusted as a custodian of this data have been significant issues in the past.

Numerous OpenStack services maintain data and metadata belonging to tenants or reference tenant information.

Tenant data stored in an OpenStack cloud may include the following items:

- Swift objects
- Compute instance ephemeral filesystem storage
- Compute instance memory
- Cinder volume data
- Public keys for Compute Access
- Virtual Machine Images in Glance
- Machine snapshots
- Data passed to OpenStack Compute's configuration-drive extension

Metadata stored by an OpenStack cloud includes the following non-exhaustive items:

- Organization name
- User's "Real Name"
- Number or size of running instances, buckets, objects, volumes, and other quota-related items
- Number of hours running instances or storing data
- IP addresses of users
- Internally generated private keys for compute image bundling

Data Disposal

OpenStack operators should strive to provide a certain level of tenant data disposal assurance. Best practices suggest that the operator sanitize cloud system media (digital and non-digital) prior to disposal, release out of organization control or release for reuse. Sanitization methods should implement an appropriate level of strength and integrity given the specific security domain and sensitivity of the information.

> "Sanitization is the process used to remove information from system media such that there is reasonable assurance that the information cannot be retrieved or reconstructed. Sanitization techniques, including clearing, purging, and destroying media information, prevent the disclosure of organizational information to unauthorized individuals when such media is reused or released for disposal." [NIST Special Publication 800-53 Revision 3]

General data disposal and sanitization guidelines as adopted from NIST recommended security controls. Cloud Operators should:

1. Track, document and verify media sanitization and disposal actions.
2. Test sanitation equipment and procedures to verify proper performance.

3. Sanitize portable, removable storage devices prior to connecting such devices to the cloud infrastructure.

4. Destroy cloud system media that cannot be sanitized.

In an OpenStack deployment you will need to address the following:

- Secure data erasure
- Instance memory scrubbing
- Block Storage volume data
- Compute instance ephemeral storage
- Bare metal server sanitization

Data not securely erased

Within OpenStack some data may be deleted, but not securely erased in the context of the NIST standards outlined above. This is generally applicable to most or all of the above-defined metadata and information stored in the database. This may be remediated with database and/or system configuration for auto vacuuming and periodic free-space wiping.

Instance memory scrubbing

Specific to various hypervisors is the treatment of instance memory. This behavior is not defined in OpenStack Compute, although it is generally expected of hypervisors that they will make a best effort to scrub memory either upon deletion of an instance, upon creation of an instance, or both.

Xen explicitly assigns dedicated memory regions to instances and scrubs data upon the destruction of instances (or domains in Xen parlance). KVM depends more greatly on Linux page management; A complex set of rules related to KVM paging is defined in the KVM documentation.

It is important to note that use of the Xen memory balloon feature is likely to result in information disclosure. We strongly recommended to avoid use of this feature.

For these and other hypervisors, we recommend referring to hypervisor-specific documentation.

Cinder volume data

Plugins to OpenStack Block Storage will store data in a variety of ways. Many plugins are specific to a vendor or technology, whereas others are more DIY solutions around filesystems such as LVM or ZFS. Methods to securely destroy data will vary from one plugin to another, from one vendor's solution to another, and from one filesystem to another.

Some backends such as ZFS will support copy-on-write to prevent data exposure. In these cases, reads from unwritten blocks will always return zero. Other backends such as LVM may not natively support this, thus the Cinder plugin takes the responsibility to override previously written blocks before handing them to users. It is important to review what assurances your chosen volume backend provides and to see what mediations may be available for those assurances not provided.

Finally, while not a feature of OpenStack, vendors and implementors may choose to add or support encryption of volumes. In this case, destruction of data is as simple as throwing away the key.

Compute instance ephemeral storage

The creation and destruction of ephemeral storage will be somewhat dependent on the chosen hypervisor and the OpenStack Compute plugin.

The libvirt plugin for compute may maintain ephemeral storage directly on a filesystem, or in LVM. Filesystem storage generally will not overwrite data when it is removed, although there is a guarantee that dirty extents are not provisioned to users.

When using LVM backed ephemeral storage, which is block-based, it is necessary that the OpenStack Compute software securely erases blocks to prevent information disclosure. There have in the past been information disclosure vulnerabilities related to improperly erased ephemeral block storage devices.

Filesystem storage is a more secure solution for ephemeral block storage devices than LVM as dirty extents cannot be provisioned

to users. However, it is important to be mindful that user data is not destroyed, so it is suggested to encrypt the backing filesystem.

Bare metal server sanitization

A bare metal server driver for Nova was under development and has since moved into a separate project called Ironic. At the time of this writing, Ironic does not appear to address sanitization of tenant data resident the physical hardware.

Additionally, it is possible for tenants of a bare metal system to modify system firmware. TPM technology, described in ##link:Management/Node Bootstrapping##, provides a solution for detecting unauthorized firmware changes.

第37章 Data Encryption

The option exists for implementors to encrypt tenant data wherever it is stored on disk or transported over a network. This is above and beyond the general recommendation that users encrypt their own data before sending it to their provider.

The importance of encrypting data on behalf of tenants is largely related to the risk assumed by a provider that an attacker could access tenant data. There may be requirements here in government, as well as requirements per-policy, in private contract, or even in case law in regard to private contracts for public cloud providers. It is recommended that a risk assessment and legal consul advised before choosing tenant encryption policies.

Per-instance or per-object encryption is preferable over, in descending order, over per-project, per-tenant, per-host, and per-cloud aggregations. This recommendation is inverse to the complexity and difficulty of implementation. Presently, in some projects it is difficult or impossible to implement encryption as loosely granular as even per-tenant. We recommend implementors make a best-effort in encrypting tenant data.

Often, data encryption relates positively to the ability to reliably destroy tenant and per-instance data, simply by throwing away the keys. It should be noted that in doing so, it becomes of great importance to destroy those keys in a reliable and secure manner.

Opportunities to encrypt data for users are present:

- Object Storage objects
- Block Storage volumes & Instance Ephemeral Filesystems
- Network data

Object Storage Objects

The ability to encrypt objects in Object Storage is presently limited to disk-level encryption per node. However, there

does exist third-party extensions and modules for per-object encryption. These modules have been proposed upstream, but have not per this writing been formally accepted. Below are some pointers:

https://github.com/Mirantis/swift-encrypt

http://www.mirantis.com/blog/on-disk-encryption-prototype-for-openstack-swift/

Block Storage Volumes & Instance Ephemeral Filesystems

The ability to encrypt volumes depends on the service backends chosen. Some backends may not support this at all.

As both block storage and compute support LVM backed storage, we can easily provide an example applicable to both systems. In deployments using LVM, encryption may be performed against the backing physical volumes. An encrypted block device would be created using the standard Linux tools, with the LVM physical volume (PV) created on top of the decrypted block device using pvcreate. Then, the vgcreate or vgmodify tool may be used to add the encrypted physical volume to an LVM volume group (VG).

A feature aimed for the Havana release provides encryption of the VM's data before it is written to disk. This allows the privacy of data to be maintained while residing on the storage device. The idea is similar to how self-encrypting drives work. This feature presents a normal block storage device to the VM but encrypts the bytes in the virtualization host before writing them to the disk. The block server operates exactly as it does when reading and writing unencrypted blocks, except special handling will be required for Block Storage features such as snapshots and live migration. Note that this feature uses an independent key manager.

Network Data

Tenant data for compute could be encrypted over IPSec or other tunnels. This is not functionality common or standard in OpenStack, but is an option available to motivated and interested implementors.

Block storage supports a variety of mechanisms for supplying mountable volumes. It is outside the scope of this guide to specify recommendations for each Block Storage backend driver. For the purpose of performance, many storage protocols are unencrypted. Some protocols such as iSCSI can provide authentication and encrypted sessions, it is our recommendation to enable these features.

第38章 Key Management

To address the often mentioned concern of tenant data privacy and limiting cloud provider liability, there is greater interest within the OpenStack community to make data encryption more ubiquitous. It is relatively easy for an end-user to encrypt their data prior to saving it to the cloud, and this is a viable path for tenant objects such as media files, database archives among others. However, when client side encryption is used for virtual machine images, block storage etc, client intervention is necessary in the form of presenting keys to unlock the data for further use. To seamlessly secure the data and yet have it accessible without burdening the client with having to manage their keys and interactively provide them calls for a key management service within OpenStack. Providing encryption and key management services as part of OpenStack eases data-at-rest security adoption, addresses customer concerns about the privacy and misuse of their data with the added advantage of limiting cloud provider liability. Provider liability is of concern in multi-tenant public clouds with respect to handing over tenant data during a misuse investigation.

A key management service is in the early stages of being developed and has a way to go before becoming an official component of OpenStack. Refer to https://github.com/cloudkeep/barbican/wiki/_pages for details.

It shall support the creation of keys, and their secure saving (with a service master-key). Some of the design questions still being debated are how much of the Key Management Interchange Protocol (KMIP) to support, key formats, and certificate management. The key manager will be pluggable to facilitate deployments that need a third-party Hardware Security Module (HSM).

OpenStack Block Storage, Cinder, is the first service looking to integrate with the key manager to provide volume encryption.

References:

- Barbican

- KMIP

第39章 Case Studies: Tenant Data

Returning to Alice and Bob, we will use this section to dive into their particular tenant data privacy requirements. Specifically, we will look into how Alice and Bob both handle tenant data, data destruction, and data encryption.

アリスのプライベートクラウド

As stated during the introduction to Alice's case study, data protection is of an extremely high priority. She needs to ensure that a compromise of one tenant's data does not cause loss of other tenant data. She also has strong regulator requirements that require documentation of data destruction activities. Alice does this using the following:

- Establishing procedures to sanitize tenant data when a program or project ends
- Track the destruction of both the tenant data and metadata via ticketing in a CMDB
- For Volume storage:
- Physical Server Issues
- To provide secure ephemeral instance storage, Alice implements qcow2 files on an encrypted filesystem.

ボブのパブリッククラウド

As stated during the introduction to Bob's case study, tenant privacy is of an extremely high priority. In addition to the requirements and actions Bob will take to isolate tenants from one another at the infrastructure layer, Bob also needs to provide assurances for tenant data privacy. Bob does this using the following:

- Establishing procedures to sanitize customer data when a customer churns
- Track the destruction of both the customer data and metadata via ticketing in a CMDB
- For Volume storage:
- Physical Server Issues
- To provide secure ephemeral instance storage, Bob implements qcow2 files on an encrypted filesystems.

第40章 Hypervisor Selection

Virtualization provides flexibility and other key benefits that enable cloud building. However, a virtualization stack also needs to be secured appropriately to reduce the risks associated with hypervisor breakout attacks. That is, while a virtualization stack can provide isolation between instances, or guest virtual machines, there are situations where that isolation can be less than perfect. Making intelligent selections for virtualization stack as well as following the best practices outlined in this chapter can be included in a layered approach to cloud security. Finally, securing your virtualization stack is critical in order to deliver on the promise of multi-tenant, either between customers in a public cloud, between business units in a private cloud, or some mixture of the two in a hybrid cloud.

In this chapter, we discuss the hypervisor selection process. In the chapters that follow, we provide the foundational information needed for securing a virtualization stack.

Hypervisors in OpenStack

Whether OpenStack is deployed within private data centers or as a public cloud service, the underlying virtualization technology provides enterprise-level capabilities in the realms of scalability, resource efficiency, and uptime. While such high-level benefits are generally available across many OpenStack-supported hypervisor technologies, there are significant differences in each hypervisor's security architecture and features, particularly when considering the security threat vectors which are unique to elastic OpenStack environments. As applications consolidate into single Infrastructure-as-a-Service (IaaS) platforms, instance isolation at the hypervisor level becomes paramount. The requirement for secure isolation holds true across commercial, government, and military communities.

Within the framework of OpenStack you can choose from any number of hypervisor platforms and corresponding OpenStack plugins to optimize your cloud environment. In the context of the OpenStack Security guide, we will be highlighting hypervisor selection

considerations as they pertain to feature sets that are critical to security. However, these considerations are not meant to be an exhaustive investigation into the pros and cons of particular hypervisors. NIST provides additional guidance in Special Publication 800-125, "Guide to Security for Full Virtualization Technologies".

Selection Criteria

As part of your hypervisor selection process, you will need to consider a number of important factors to help increase your security posture. Specifically, we will be looking into the following areas:

- Team Expertise
- Product or Project maturity
- Certifications, Attestations
- Additional Security Features
- Hypervisor vs. Baremetal
- Hardware Concerns
- Common Criteria

Additionally, the following security-related criteria are highly encouraged to be evaluated when selecting a hypervisor for OpenStack deployments:

- Has the hypervisor undergone Common Criteria certification? If so, to what levels?
- Is the underlying cryptography certified by a third-party?

Team Expertise

Most likely, the most important aspect in hypervisor selection is the expertise of your staff in managing and maintaining a particular hypervisor platform. The more familiar your team is with a given product, its configuration, and its eccentricities, the less likely will there be configuration mistakes. Additionally, having staff expertise spread across an organization on a given hypervisor will increase availability of

your systems, allow for developing a segregation of duties, and mitigate problems in the event that a team member is unavailable.

Product or Project Maturity

The maturity of a given hypervisor product or project is critical to your security posture as well. Product maturity will have a number of effects once you have deployed your cloud, in the context of this security guide we are interested in the following:

- Availability of expertise
- Active developer and user communities
- Timeliness and Availability of updates
- Incidence response

One of the biggest indicators of a hypervisor's maturity is the size and vibrancy of the community that surrounds it. As this concerns security, the quality of the community will affect the availability of expertise should you need additional cloud operators. It is also a sign of how widely deployed the hypervisor is, in turn leading to the battle readiness of any reference architectures and best practices.

Further, the quality of community, as it surrounds an open source hypervisor like KVM or Xen, will have a direct impact on the timeliness of bug fixes and security updates. When investigating both commercial and open source hypervisors, you will want to look into their release and support cycles as well as the time delta between the announcement of a bug or security issue and a patch or response. Lastly, the supported capabilities of OpenStack compute vary depending on the hypervisor chosen. Refer to the OpenStack Hypervisor Support Matrix for OpenStack compute feature support by hypervisor.

Certifications and Attestations

One additional consideration when selecting a hypervisor is the availability of various formal certifications and attestations. While they may not be requirements for your specific organization, these certifications and attestations speak to the maturity, production readiness, and thoroughness of

the testing a particular hypervisor platform has been subjected to.

Common Criteria

Common Criteria is an internationally standardized software evaluation process, used by governments and commercial companies to validate software technologies perform as advertised. In the government sector, NSTISSP No. 11 mandates that U.S. Government agencies only procure software which has been Common Criteria certified, a policy which has been in place since July 2002. It should be specifically noted that OpenStack has not undergone Common Criteria certification, however many of the available hypervisors have.

In addition to validating a technologies capabilities, the Common Criteria process evaluates how technologies are developed.

- How is source code management performed?
- How are users granted access to build systems?
- Is the technology cryptographically signed before distribution?

The KVM hypervisor has been Common Criteria certified through the U.S. Government and commercial distributions, which have been validated to separate the runtime environment of virtual machines from each other, providing foundational technology to enforce instance isolation. In addition to virtual machine isolation, KVM has been Common Criteria certified to

> "provide system-inherent separation mechanisms to the resources of virtual machines. This separation ensures that large software component used for virtualizing and simulating devices executing for each virtual machine cannot interfere with each other. Using the SELinux multi-category mechanism, the virtualization and simulation software instances are isolated. The virtual machine management framework configures SELinux multi-category settings transparently to the administrator"

While many hypervisor vendors, such as Red Hat, Microsoft, and VMWare have achieved Common Criteria Certification their underlying certified feature set differs. It is recommended

to evaluate vendor claims to ensure they minimally satisfy the following requirements:

Identification and Authentication	Identification and authentication using pluggable authentication modules (PAM) based upon user passwords. The quality of the passwords used can be enforced through configuration options.
Audit	The system provides the capability to audit a large number of events including individual system calls as well as events generated by trusted processes. Audit data is collected in regular files in ASCII format. The system provides a program for the purpose of searching the audit records. The system administrator can define a rule base to restrict auditing to the events they are interested in. This includes the ability to restrict auditing to specific events, specific users, specific objects or a combination of all of this. Audit records can be transferred to a remote audit daemon.
Discretionary Access Control	Discretionary Access Control (DAC) restricts access to file system objects based on Access Control Lists (ACLs) that include the standard UNIX permissions for user, group and others. Access control mechanisms also protect IPC objects from unauthorized access. The system includes the ext4 file system, which supports POSIX ACLs. This allows defining access rights to files within this type of file system down to the granularity of a single user.
Mandatory Access Control	Mandatory Access Control (MAC) restricts access to objects based on labels assigned to subjects and objects. Sensitivity labels are automatically attached to processes and objects. The access control policy enforced using these labels is derived from

	the BellLaPadula access control model. SELinux categories are attached to virtual machines and its resources. The access control policy enforced using these categories grant virtual machines access to resources if the category of the virtual machine is identical to the category of the accessed resource. The TOE implements non-hierarchical categories to control access to virtual machines.
Role-Based Access Control	Role-based access control (RBAC) allows separation of roles to eliminate the need for an all-powerful system administrator.
Object Reuse	File system objects as well as memory and IPC objects will be cleared before they can be reused by a process belonging to a different user.
セキュリティ管理	The management of the security critical parameters of the system is performed by administrative users. A set of commands that require root privileges (or specific roles when RBAC is used) are used for system management. Security parameters are stored in specific files that are protected by the access control mechanisms of the system against unauthorized access by users that are not administrative users.
Secure Communication	The system supports the definition of trusted channels using SSH. Password based authentication is supported. Only a restricted number of cipher suites are supported for those protocols in the evaluated configuration.
Storage Encryption	The system supports encrypted block devices to provide storage confidentiality via dm_crypt.
TSF Protection	While in operation, the kernel software and data are protected by the hardware memory protection mechanisms. The memory and process management components of

	the kernel ensure a user process cannot access kernel storage or storage belonging to other processes. Non-kernel TSF software and data are protected by DAC and process isolation mechanisms. In the evaluated configuration, the reserved user ID root owns the directories and files that define the TSF configuration. In general, files and directories containing internal TSF data, such as configuration files and batch job queues, are also protected from reading by DAC permissions. The system and the hardware and firmware components are required to be physically protected from unauthorized access. The system kernel mediates all access to the hardware mechanisms themselves, other than program visible CPU instruction functions. In addition, mechanisms for protection against stack overflow attacks are provided.

Cryptography Standards

Several cryptography algorithms are available within OpenStack for identification and authorization, data transfer and protection of data at rest. When selecting a hypervisor, the following are recommended algorithms and implementation standards to ensure the virtualization layer supports:

Algorithm	Key Length	Intended Purpose	Security Function	Implementation Standard
AES	128 bits, 192 bits, 256 bits	Encryption / Decryption	Protected Data Transfer, Protection for Data at Rest	RFC 4253
TDES	168 bits	Encryption / Decryption	Protected Data Transfer	RFC 4253
RSA	1024 bits, 2048 bits,	Authentication, Key Exchange	Identification and Authentication	U.S. NIST FIPS PUB 186-3

	3072 bits		Protected Data Transfer	
DSA	L=1024,N=160 bits	Authentication, Key Exchange	Identification and Authentication, Protected Data Transfer	U.S. NIST FIPS PUB 186-3
Serpent	128, 196, or256 bit	Encryption / Decryption	Protection of Data at Rest	http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf
Twofish	128, 196, or256 bit	Encryption / Decryption	Protection of Data at Rest	http://www.schneier.com/paper-twofish-paper.html
SHA-1	-	MessageDigest	Protection of Data at Rest, Protected Data Transfer	U.S. NIST FIPS 180-3
SHA-2(224-, 256-, 384-, 512 bit)	-	MessageDigest	Protection for Data at Rest, Identification and Authentication	U.S. NIST FIPS 180-3

FIPS 140-2

In the United States the National Institute of Science and Technology (NIST) certifies cryptographic algorithms through a process known the Cryptographic Module Validation Program. NIST certifies algorithms for conformance against Federal Information Processing Standard 140-2 (FIPS 140-2), which ensures:

> Products validated as conforming to FIPS 140-2 are accepted by the Federal agencies of both countries [United States and Canada] for the protection of sensitive information (United States) or Designated Information (Canada). The goal of the CMVP is to promote the use of validated cryptographic modules and provide Federal agencies with a security metric to use in procuring equipment containing validated cryptographic modules.

When evaluating base hypervisor technologies, consider if the hypervisor has been certified against FIPS 140-2. Not only is conformance against FIPS 140-2 mandated per U.S. Government policy, formal certification indicates that a given implementation of a cryptographic algorithm has been reviewed for conformance against module specification, cryptographic module ports and interfaces; roles, services, and authentication; finite state model; physical security; operational environment; cryptographic key management; electromagnetic interference/ electromagnetic compatibility (EMI/EMC); self-tests; design assurance; and mitigation of other attacks.

Hardware Concerns

Further, when evaluating a hypervisor platform the supportability of the hardware the hypervisor will run on should be considered. Additionally, consider the additional features available in the hardware and how those features are supported by the hypervisor you chose as part of the OpenStack deployment. To that end, hypervisors will each have their own hardware compatibility lists (HCLs). When selecting compatible hardware it is important to know in advance which hardware-based virtualization technologies are important from a security perspective.

説明	Technology	Explanation
I/O MMU	VT-d / AMD-Vi	Required for protecting PCI-passthrough
Intel Trusted Execution Technology	Intel TXT / SEM	Required for dynamic attestation services
PCI-SIG I/O virtualization	SR-IOV, MR-IOV, ATS	Required to allow secure sharing of PCI Express devices
Network virtualization	VT-c	Improves performance of network I/O on hypervisors

Hypervisor vs. Baremetal

To wrap up our discussion around hypervisor selection, it is important to call out the differences between using LXC (Linux Containers) or Baremetal systems vs using a hypervisor like KVM. Specifically, the focus of this security guide will be largely based on having a hypervisor and virtualization

platform. However, should your implementation require the use of a baremetal or LXC environment, you will want to pay attention to the particular differences in regard to deployment of that environment. In particular, you will need to provide your end users with assurances that the node has been properly sanitized of their data prior to re-provisioning. Additionally, prior to reusing a node, you will need to provide assurances that the hardware has not been tampered or otherwise compromised.

It should be noted that while OpenStack has a baremetal project, a discussion of the particular security implications of running baremetal is beyond the scope of this book.

Finally, due to the time constraints around a book sprint, the team chose to use KVM as the hypervisor in our example implementations and architectures.

注記

There is an OpenStack Security Note pertaining to the use of LXC in Nova.

Additional Security Features

Another thing to look into when selecting a hypervisor platform is the availability of specific security features. In particular, we are referring to features like Xen Server's XSM or Xen Security Modules, sVirt, Intel TXT, and AppArmor. The presence of these features will help increase your security profile as well as provide a good foundation.

The following table calls out these features by common hypervisor platforms.

	KSM	XSM	sVirt	TXT	AppArmor	cGroups	MAC Policy
KVM	X		X	X	x	x	x
Xen		X		X			x
ESXi				X			
Hyper-V							

KSM: Kernel Samepage Merging

XSM: Xen Security Modules

xVirt: Mandatory Access Control for Linux-based virtualization

TXT: Intel Trusted Execution Technology

AppArmor: Linux security module implementing MAC

cgroups: Linux kernel feature to control resource usage

MAC Policy: Mandatory Access Control; may be implemented with SELinux or other operating systems

* Features in this table may not be applicable to all hypervisors or directly mappable between hypervisors.

第41章 Hardening the Virtualization Layers

In the beginning of this chapter we discuss the use of both physical and virtual hardware by instances, the associated security risks, and some recommendations for mitigating those risks. We conclude the chapter with a discussion of sVirt, an open source project for integrating SELinux mandatory access controls with the virtualization components.

Physical Hardware (PCI Passthrough)

Many hypervisors offer a functionality known as PCI passthrough. This allows an instance to have direct access to a piece of hardware on the node. For example, this could be used to allow instances to access video cards offering the compute unified device architecture (CUDA) for high performance computation. This feature carries two types of security risks: direct memory access and hardware infection.

Direct memory access (DMA) is a feature that permits certain hardware devices to access arbitrary physical memory addresses in the host computer. Often video cards have this capability. However, an instance should not be given arbitrary physical memory access because this would give it full view of both the host system and other instances running on the same node. Hardware vendors use an input/output memory management unit (IOMMU) to manage DMA access in these situations. Therefore, cloud architects should ensure that the hypervisor is configured to utilize this hardware feature.

- KVM: How to assign devices with VT-d in KVM
- Xen: VTd Howto

注記

The IOMMU feature is marketed as VT-d by Intel and AMD-Vi by AMD.

A hardware infection occurs when an instance makes a malicious modification to the firmware or some other part of a device. As this device is used by other instances, or even the host OS, the malicious code can spread into these systems. The end result is that one instance can run code outside of its security domain. This is a potential problem in any hardware sharing scenario. The problem is specific to this scenario because it is harder to reset the state of physical hardware than virtual hardware.

Solutions to the hardware infection problem are domain specific. The strategy is to identify how an instance can modify hardware state then determine how to reset any modifications when the instance is done using the hardware. For example, one option could be to re-flash the firmware after use. Clearly there is a need to balance hardware longevity with security as some firmwares will fail after a large number of writes. TPM technology, described in link:Management/Node Bootstrapping, provides a solution for detecting unauthorized firmware changes. Regardless of the strategy selected, it is important to understand the risks associated with this kind of hardware sharing so that they can be properly mitigated for a given deployment scenario.

Additionally, due to the risk and complexities associated with PCI passthrough, it should be disabled by default. If enabled for a specific need, you will need to have appropriate processes in place to ensure the hardware is clean before re-issue.

Virtual Hardware (QEMU)

When running a virtual machine, virtual hardware is a software layer that provides the hardware interface for the virtual machine. Instances use this functionality to provide network, storage, video, and other devices that may be needed. With this in mind, most instances in your environment will exclusively use virtual hardware, with a minority that will require direct hardware access. The major open source hypervisors use QEMU for this functionality. While QEMU fills an important need for virtualization platforms, it has proven to be a very challenging

software project to write and maintain. Much of the functionality in QEMU is implemented with low-level code that is difficult for most developers to comprehend. Furthermore, the hardware virtualized by QEMU includes many legacy devices that have their own set of quirks. Putting all of this together, QEMU has been the source of many security problems, including hypervisor breakout attacks.

For the reasons stated above, it is important to take proactive steps to harden QEMU. We recommend three specific steps: minimizing the code base, using compiler hardening, and using mandatory access controls, such as sVirt, SELinux, or AppArmor.

Minimizing the Qemu Code base

One classic security principle is to remove any unused components from your system. QEMU provides support for many different virtual hardware devices. However, only a small number of devices are needed for a given instance. Most instances will use the virtio devices. However, some legacy instances will need access to specific hardware, which can be specified using glance metadata:

```
glance image-update ¥
    --property hw_disk_bus=ide ¥
    --property hw_cdrom_bus=ide ¥
    --property hw_vif_model=e1000 ¥
    f16-x86_64-openstack-sda
```

A cloud architect should decide what devices to make available to cloud users. Anything that is not needed should be removed from QEMU. This step requires recompiling QEMU after modifying the options passed to the QEMU configure script. For a complete list of up-to-date options simply run ./configure --help from within the QEMU source directory. Decide what is needed for your deployment, and disable the remaining options.

Compiler Hardening

The next step is to harden QEMU using compiler hardening options. Modern compilers provide a variety of compile time options to improve the security of the resulting binaries. These features, which we will describe in more detail below, include relocation read-only (RELRO), stack canaries, never execute (NX),

position independent executable (PIE), and address space layout randomization (ASLR).

Many modern linux distributions already build QEMU with compiler hardening enabled, so you may want to verify your existing executable before proceeding with the information below. One tool that can assist you with this verification is called checksec.sh.

- RELocation Read-Only (RELRO): Hardens the data sections of an executable. Both full and partial RELRO modes are supported by gcc. For QEMU full RELRO is your best choice. This will make the global offset table read-only and place various internal data sections before the program data section in the resulting executable.
- Stack Canaries: Places values on the stack and verifies their presence to help prevent buffer overflow attacks.
- Never eXecute (NX): Also known as Data Execution Prevention (DEP), ensures that data sections of the executable can not be executed.
- Position Independent Executable (PIE): Produces a position independent executable, which is necessary for ASLR.
- Address Space Layout Randomization (ASLR) : This ensures that both code and data regions will be randomized. Enabled by the kernel (all modern linux kernels support ASLR), when the executable is built with PIE.

Putting this all together, and adding in some additional useful protections, we recommend the following compiler options for gcc when compiling QEMU:

```
CFLAGS="-arch x86_64 -fstack-protector-all -Wstack-protector --param ssp-
buffer-size=4 -pie -fPIE -ftrapv -D_FORTIFY_SOURCE=2 O2 -Wl,-z,relro,-z,now"
```

We recommend testing your QEMU executable file after it is compiled to ensure that the compiler hardening worked properly.

Most cloud deployments will not want to build software such as QEMU by hand. It is better to use packaging to ensure that the process is repeatable and to ensure that the end result can be easily deployed throughout the cloud. The references below provide some additional details on applying compiler hardening options to existing packages.

- DEB packages: Hardening Walkthrough
- RPM packages: How to create an RPM package

強制アクセス制御

Compiler hardening makes it more difficult to attack the QEMU process. However, if an attacker does succeed, we would like to limit the impact of the attack. Mandatory access controls accomplish this by restricting the privileges on QEMU process to only what is needed. This can be accomplished using sVirt / SELinux or AppArmor. When using sVirt, SELinux is configured to run every QEMU process under a different security context. AppArmor can be configured to provide similar functionality. We provide more details on sVirt in the instance isolation section below.

sVirt: SELinux + Virtualization

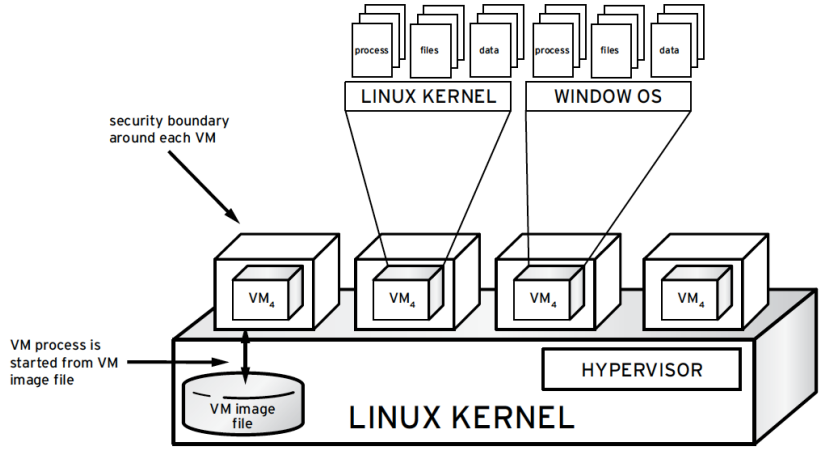
With unique kernel-level architecture and National Security Agency (NSA) developed security mechanisms, KVM provides foundational isolation technologies for multi tenancy.
With developmental origins dating back to 2002, the Secure Virtualization (sVirt) technology is the application of SELinux against modern day virtualization. SELinux, which was designed to apply separation control based upon labels, has been extended to provide isolation between virtual machine processes, devices, data files and system processes acting upon their behalf.

OpenStack's sVirt implementation aspires to protect hypervisor hosts and virtual machines against two primary threat vectors:

- Hypervisor threats A compromised application running within a virtual machine attacks the hypervisor to access underlying resources. For example, the host OS, applications, or devices within the physical machine. This is a threat vector unique to virtualization and represents considerable risk as the underlying real machine can be compromised due to vulnerability in a single virtual application.
- Virtual Machine (multi-tenant) threats A compromised application running within a VM attacks the hypervisor to access/control another virtual machine and its resources. This

is a threat vector unique to virtualization and represents considerable risk as a multitude of virtual machine file images could be compromised due to vulnerability in a single application. This virtual network attack is a major concern as the administrative techniques for protecting real networks do not directly apply to the virtual environment.

Each KVM-based virtual machine is a process which is labeled by SELinux, effectively establishing a security boundary around each virtual machine. This security boundary is monitored and enforced by the Linux kernel, restricting the virtual machine's access to resources outside of its boundary such as host machine data files or other VMs.



As shown above, sVirt isolation is provided regardless of the guest Operating System running inside the virtual machine -- Linux or Windows VMs can be used. Additionally, many Linux distributions provide SELinux within the operating system, allowing the virtual machine to protect internal virtual resources from threats.

Labels and Categories

KVM-based virtual machine instances are labelled with their own SELinux data type, known as svirt_image_t. Kernel level protections prevent unauthorized system processes, such as malware, from manipulating the virtual machine image files on disk. When virtual machines are powered off, images are stored as svirt_image_t as shown below:

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

The svirt_image_t label uniquely identifies image files on disk, allowing for the SELinux policy to restrict access. When a KVM-based Compute image is powered on, sVirt appends a random numerical identifier to the image. sVirt is technically capable of assigning numerical identifiers to 524,288 virtual machines per hypervisor node, however OpenStack deployments are highly unlikely to encounter this limitation.

This example shows the sVirt category identifier:

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

Booleans

To ease the administrative burden of managing SELinux, many enterprise Linux platforms utilize SELinux Booleans to quickly change the security posture of sVirt.

Red Hat Enterprise Linux-based KVM deployments utilize the following sVirt booleans:

sVirt SELinux Boolean	Description
virt_use_common	Allow virt to use serial/parallel communication ports.
virt_use_fusefs	Allow virt to read FUSE mounted files.
virt_use_nfs	Allow virt to manage NFS mounted files.
virt_use_samba	Allow virt to manage CIFS mounted files.
virt_use_sanlock	Allow confined virtual guests to interact with the sanlock.
virt_use_sysfs	Allow virt to manage device configuration (PCI).
virt_use_usb	Allow virt to use USB devices.
virt_use_xserver	Allow virtual machine to interact with the X Window System.

第42章 Case Studies: Instance Isolation

In this case study we discuss how Alice and Bob would ensure that their instances are properly isolated. First we consider hypervisor selection, and then techniques for hardening QEMU and applying mandatory access controls.

アリスのプライベートクラウド

Alice chooses Xen for the hypervisor in her cloud due to a strong internal knowledge base and a desire to use the Xen security modules (XSM) for fine-grained policy enforcement.

Alice is willing to apply a relatively large amount of resources to software packaging and maintenance. She will use these resources to build a highly customized version of QEMU that has many components removed, thereby reducing the attack surface. She will also ensure that all compiler hardening options are enabled for QEMU. Alice accepts that these decisions will increase long-term maintenance costs.

Alice writes XSM policies (for Xen) and SELinux policies (for Linux domain 0, and device domains) to provide stronger isolation between the instances. Alice also uses the Intel TXT support in Xen to measure the hypervisor launch in the TPM.

ボブのパブリッククラウド

Bob is very concerned about instance isolation since the users in a public cloud represent anyone with a credit card, meaning they are inherently untrusted. Bob has just started hiring the team that will deploy the cloud, so he can tailor his candidate search for specific areas of expertise. With this in mind, Bob chooses a hypervisor based on its technical features, certifications, and community support. KVM has an EAL 4+ common criteria rating, with a layered security protection profile (LSPP) to provide

added assurance for instance isolation. This, combined with the strong support for KVM within the OpenStack community drives Bob's decision to use KVM.

Bob weighs the added cost of repackaging QEMU and decides that he cannot commit those resources to the project. Fortunately, his Linux distribution has already enabled the compiler hardening options. So he decides to use this QEMU package. Finally, Bob leverages sVirt to manage the SELinux polices associated with the virtualization stack.

第43章 Security Services for Instances

One of the virtues of running instances in a virtualized environment is that it opens up new opportunities for security controls that are not typically available when deploying onto bare metal. There are several technologies that can be applied to the virtualization stack that bring improved information assurance for cloud tenants.

Deployers or users of OpenStack with strong security requirements may want to consider deploying these technologies. Not all are applicable in every situation, indeed in some cases technologies may be ruled out for use in a cloud because of prescriptive business requirements. Similarly some technologies inspect instance data such as run state which may be undesirable to the users of the system.

In this chapter we explore these technologies and describe the situations where they can be used to enhance security for instances or underlying instances. We also seek to highlight where privacy concerns may exist. These include data pass through, introspection, or providing a source of entropy. In this section we highlight the following additional security services:

- Entropy to Instances
- Scheduling Instances to Nodes
- Trusted Images
- Instance Migrations

Entropy To Instances

We consider entropy to refer to the quality and source of random data that is available to an instance. Cryptographic

technologies typically rely heavily on randomness, requiring a high quality pool of entropy to draw from. It is typically hard for a virtual machine to get enough entropy to support these operations. Entropy starvation can manifest in instances as something seemingly unrelated for example, slow boot times because the instance is waiting for ssh key generation. Entropy starvation may also motivate users to employ poor quality entropy sources from within the instance, making applications running in the cloud less secure overall.

Fortunately, a cloud architect may address these issues by providing a high quality source of entropy to the cloud instances. This can be done by having enough hardware random number generators (HRNG) in the cloud to support the instances. In this case, "enough" is somewhat domain specific. For everyday operations, a modern HRNG is likely to produce enough entropy to support 50-100 compute nodes. High bandwidth HRNGs, such as the RdRand instruction available with Intel Ivy Bridge and newer processors could potentially handle more nodes. For a given cloud, an architect needs to understand the application requirements to ensure that sufficient entropy is available.

Once the entropy is available in the cloud, the next step is getting that entropy into the instances. Tools such as the entropy gathering daemon (EGD) provide a way to fairly and securely distribute entropy through a distributed system. Support exists for using the EGD as an entropy source for LibVirt.

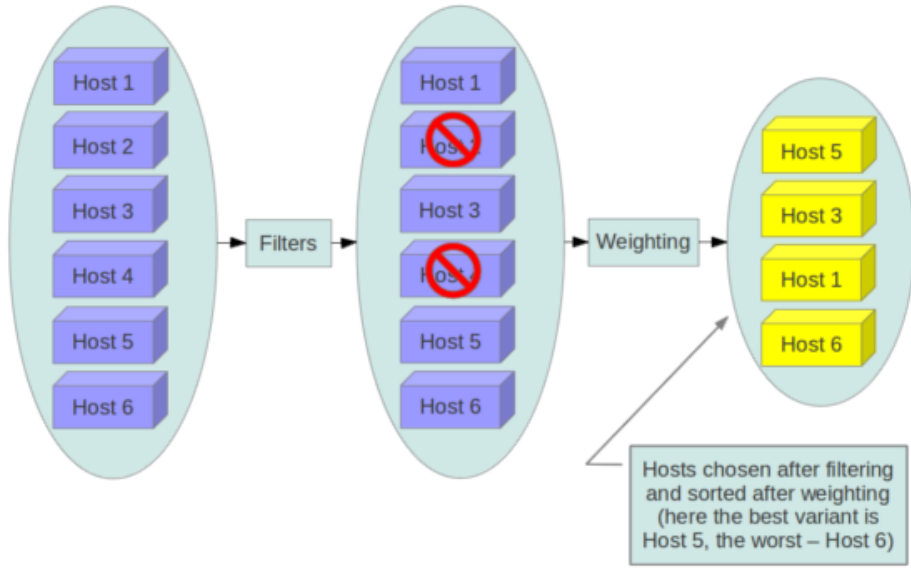
Compute support for these features is not generally available, but it would only require a moderate amount of work for implementors to integrate this functionality.

Scheduling Instances to Nodes

Before an instance is created, a host for the image instantiation must be selected. This selection is performed by the nova-scheduler which determines how to dispatch compute and volume requests.

The default nova scheduler in Grizzly is the Filter Scheduler, although other schedulers exist (see the section Scheduling in the OpenStack Configuration Reference). The filter scheduler works in collaboration with 'filters' to decide where an

instance should be started. This process of host selection allows administrators to fulfill many different security requirements. Depending on the cloud deployment type for example, one could choose to have tenant instances reside on the same hosts whenever possible if data isolation was a primary concern, conversely one could attempt to have instances for a tenant reside on as many different hosts as possible for availability or fault tolerance reasons. The following diagram demonstrates how the filter scheduler works:



The use of scheduler filters may be used to segregate customers, data, or even discard machines of the cloud that cannot be attested as secure. This generally applies to all OpenStack projects offering a scheduler. When building a cloud, you may choose to implement scheduling filters for a variety of security-related purposes.

Below we highlight a few of the filters that may be useful in a security context, depending on your requirements, the full set of filter documentation is documented in the Filter Scheduler section of the OpenStack Configuration Reference.

Tenant Driven Whole Host Reservation

There currently exists a blueprint for whole host reservation - This would allow a tenant to exclusively reserve hosts for only it's instances, incurring extra costs.

ホストアグリゲート

While not a filter in themselves, host aggregates allow administrators to assign key-value pairs to groups of machines. This allows cloud administrators, not users, to partition up their compute host resources. Each node can have multiple aggregates (see the Host Aggregates section of the OpenStack Configuration Reference for more information on creating and managing aggregates).

AggregateMultiTenancyIsolation

Isolates tenants to specific host aggregates. If a host is in an aggregate that has the metadata key filter_tenant_id it will only create instances from that tenant (or list of tenants). A host can be in multiple aggregates. If a host does not belong to an aggregate with the metadata key, it can create instances from all tenants.

DifferentHostFilter

Schedule the instance on a different host from a set of instances. To take advantage of this filter, the requester must pass a scheduler hint, using different_host as the key and a list of instance uuids as the value. This filter is the opposite of the SameHostFilter.

GroupAntiAffinityFilter

The GroupAntiAffinityFilter ensures that each instance in a group is on a different host. To take advantage of this filter, the requester must pass a scheduler hint, using group as the key and a list of instance uuids as the value.

Trusted Compute Pools

There exists a scheduler filter which integrates with the Open Attestation Project (OATS) to define scheduler behavior according

to the attestation of PCRs received from a system using Intel TXT.

It is unclear if this feature is compatible with AMD's similar SEM, although the OpenAttestation agent relies on the vendor-agnostic TrouSerS library.

Trusted Images

With regards to images, users will be working with pre-installed images or images that they upload themselves. In both cases, users will want to ensure that the image they are ultimately running has not been tampered with. This requires some source of truth such as a checksum for the known good version of an image as well as verification of the running image. This section describes the current best practices around image handling, while also calling out some of the existing gaps in this space.

Image Creation Process

The OpenStack Documentation provides guidance on how to create and upload an image to Glance. Additionally it is assumed that you have a process by which you install and harden operating systems. Thus, the following items will provide additional guidance on how to ensure your images are built securely prior to upload. There are a variety of options for obtaining images. Each has specific steps that help validate the image's provenance.

The first option is to obtain boot media from a trusted source.

```
mkdir -p /tmp/download_directorycd /tmp/download_directory

wget http://mirror.anl.gov/pub/ubuntu-iso/CDs/precise/ubuntu-12.04.2-server-
amd64.iso
wget http://mirror.anl.gov/pub/ubuntu-iso/CDs/precise/SHA256SUMS
wget http://mirror.anl.gov/pub/ubuntu-iso/CDs/precise/SHA256SUMS.gpg
gpg --keyserver hkp://keyserver.ubuntu.com --recv-keys 0xFBB75451
gpg --verify SHA256SUMS.gpg SHA256SUMSsha256sum -c SHA256SUMS 2>&1 | grep OK
```

The second option is to use the OpenStack Virtual Machine Image Guide. In this case, you will want to follow your organizations OS hardening guidelines or those provided by a trusted third-party such as the RHEL6 STIG.

The final option is to use an automated image builder. The following example uses the Oz image builder. The OpenStack community has recently created a newer tool worth investigating: disk-image-builder. We have not evaluated this tool from a security perspective.

Example of RHEL 6 CCE-26976-1 which will help implement NIST 800-53 Section AC-19(d) in Oz.

```
<template>
<name>centos64</name>
<os>
  <name>RHEL-6</name>
  <version>4</version>
  <arch>x86_64</arch>
  <install type='iso'>
  <iso>http://trusted_local_iso_mirror/isos/x86_64/RHEL-6.4-x86_64-bin-DVD1.
iso</iso>
  </install>
  <rootpw>CHANGE THIS TO YOUR ROOT PASSWORD</rootpw>
</os>
<description>RHEL 6.4 x86_64</description>
<repositories>
  <repository name='epel-6'>
    <url>http://download.fedoraproject.org/pub/epel/6/$basearch</url>
    <signed>no</signed>
  </repository>
</repositories>
<packages>
  <package name='epel-release'/>
  <package name='cloud-utils'/>
  <package name='cloud-init'/>
</packages>
<commands>
  <command name='update'>
  yum update
  yum clean all
  sed -i '/^HWADDR/d' /etc/sysconfig/network-scripts/ifcfg-eth0
  echo -n > /etc/udev/rules.d/70-persistent-net.rules
  echo -n > /lib/udev/rules.d/75-persistent-net-generator.rules
  chkconfig --level 0123456 autofs off
  service autofs stop
  </command>
</commands>
</template>
```

Note, it is the recommendation of this guide to shy away from the manual image building process as it is complex and prone to error. Further, using an automated system like Oz or disk-image-

builder for image building, or a configuration management utility like Chef or Puppet for post boot image hardening gives you the ability to produce a consistent image as well as track compliance of your base image to its respective hardening guidelines over time.

If subscribing to a public cloud service, you should check with the cloud provider for an outline of the process used to produce their default images. If the provider allows you to upload your own images, you will want to ensure that you are able to verify that your image was not modified before you spin it up. To do this, refer to the following section on Image Provenance.

Image Provenance and Validation

Unfortunately, it is not currently possible to force Compute to validate an image hash immediately prior to starting an instance. To understand the situation, we begin with a brief overview of how images are handled around the time of image launch.

Images come from the glance service to the nova service on a node. This transfer should be protected by running over SSL. Once the image is on the node, it is verified with a basic checksum and then it's disk is expanded based on the size of the instance being launched. If, at a later time, the same image is launched with the same instance size on this node, it will be launched from the same expanded image. Since this expanded image is not re-verified before launching, it could be tampered with and the user would not have any way of knowing, beyond a manual inspection of the files in the resulting image.

We hope that future versions of Compute and/or the Image Service will offer support for validating the image hash before each instance launch. An alternative option that would be even more powerful would be allow users to sign an image and then have the signature validated when the instance is launched.

Instance Migrations

OpenStack and the underlying virtualization layers provide for the live migration of images between OpenStack nodes allowing you to seamlessly perform rolling upgrades of your OpenStack compute nodes without instance downtime. However, live migrations

also come with their fair share of risk. To understand the risks involved, it is important to first understand how a live migration works. The following are the high level steps preformed during a live migration.

1. Start instance on destination host

2. Transfer memory

3. Stop the guest & sync disks

4. Transfer state

5. Start the guest

Live Migration Risks

At various stages of the live migration process the contents of an instances run time memory and disk are transmitted over the network in plain text. Thus there are several risks that need to be addressed when using live migration. The following in-exhaustive list details some of these risks:

- Denial of Service (DoS) : If something fails during the migration process, the instance could be lost.
- Data Exposure : Memory or disk transfers must be handled securely.
- Data Manipulation : If memory or disk transfers are not handled securely, then an attacker could manipulate user data during the migration.
- Code Injection : If memory or disk transfers are not handled securely, then an attacker could manipulate executables, either on disk or in memory, during the migration.

Live Migration Mitigations

There are several methods to mitigate some of the risk associated with live migrations, the following list details some of these:

- Disable Live Migration
- Isolated Migration Network

- Encrypted Live Migration

Disable Live Migration

At this time, live migration is enabled in OpenStack by default. Live migrations can be disabled by adding the following lines to the nova policy.json file:

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

Migration Network

As a general practice, live migration traffic should be restricted to the management security domain. Indeed live migration traffic, due to its plain text nature and the fact that you are transferring the contents of disk and memory of a running instance, it is recommended you further separate live migration traffic onto a dedicated network. Isolating the traffic to a dedicated network can reduce the risk of exposure.

Encrypted Live Migration

If your use case involves keeping live migration enabled, then libvirtd can provide tunneled, encrypted live migrations. That said, this feature is not currently exposed in OpenStack Dashboard, nor the nova-client commands and can only be accessed through manual configuration of libvritd. Encrypted live migration modifies the live migration process by first copying the instance data from the running hypervisor to libvirtd. From there an encrypted tunnel is created between the libvirtd processes on both hosts. Finally, the destination libvirtd process copies the instance back to the underlying hypervisor.

第44章 Case Studies: Instance Management

In this case study we discuss how Alice and Bob would architect their clouds with respect to instance entropy, scheduling instances, trusted images, and instance migrations.

アリスのプライベートクラウド

Alice has a need for lots of high quality entropy in the instances. For this reason, she decides to purchase hardware with Intel Ivy Bridge chip sets that support the RdRand instruction on each compute node. Using the entropy gathering daemon (EGD) and LibVirt's EGD support, Alice ensures that this entropy pool is distributed to the instances on each compute node.

For instance scheduling, Alice uses the trusted compute pools to ensure that all cloud workloads are deployed to nodes that presented a proper boot time attestation. Alice decides to disable user permissions for image uploading to help ensure that the images used in the cloud are generated in a known and trusted manner by the cloud administrators.

Finally, Alice disables instance migrations as this feature is less critical for the high performance application workloads expected to run in this cloud. This helps avoid the various security concerns related to instance migrations.

ボブのパブリッククラウド

Bob is aware that entropy will be a concern for some of his customers, such as those in the financial industry. However, due to the added cost and complexity, Bob has decided to forgo integrating hardware entropy into the first iteration of his cloud. He adds hardware entropy as a fast-follow to do for a later improvement for the second generation of his cloud architecture.

Bob is interested in ensuring that customers receive a high quality of service. He is concerned that providing too much explicit user control over instance scheduling could negatively impact the quality of service. So he disables this feature. Bob provides images in the cloud from a known trusted source for users to use. Additionally, he also allows users to upload their own images. However, users cannot generally share their images. This helps prevent a user from sharing a malicious image, which could negatively impact the security of other users in the cloud.

For migrations, Bob wants to enable secure instance migrations in order to support rolling upgrades with minimal user downtime. Bob ensures that all migrations occur on an isolated VLAN. He plans to defer implementing encrypted migrations until this is better supported in Nova client tools. However, he makes a note to track this carefully and switch to encrypted migrations as soon as possible.

第45章 Forensics and Incident Response

A lot of activity goes on within a cloud environment. It is a mix of hardware, operating systems, virtual machine managers, the OpenStack services, cloud-user activity such as creating instances and attaching storage, the network underlying the whole, and finally end-users using the applications running on the various instances.

The generation and collection of logs is an important component of securely monitoring an OpenStack infrastructure. Logs provide visibility into the day-to-day actions of administrators, tenants, and guests, in addition to the activity in the compute, networking, and storage and other components that comprise your OpenStack deployment.

The basics of logging: configuration, setting log level, location of the log files, and how to use and customize logs, as well as how to do centralized collections of logs is well covered in the OpenStack Operations Guide.

Logs are not only valuable for proactive security and continuous compliance activities, but they are also a valuable information source for investigating and responding to incidents.

For instance, analyzing the access logs of Identity Service or its replacement authentication system would alert us to failed logins, their frequency, origin IP, whether the events are restricted to select accounts etc. Log analysis supports detection.

On detection, further action may be to black list an IP, or recommend strengthening user passwords, or even de-activating a user account if it is deemed dormant.

Monitoring Use Cases

Monitoring events is more pro-active and provides real-time detection and response. There are several tools to aid in monitoring.

In the case of an OpenStack cloud instance, we need to monitor the hardware, the OpenStack services, and the cloud resource usage. The last stems from wanting to be elastic, to scale to the dynamic needs of the users.

Here are a few important use cases to consider when implementing log aggregation, analysis and monitoring. These use cases can be implemented and monitored through various commercial and open source tools, homegrown scripts, etc. These tools and scripts can generate events that can then be sent to the administrators through email or integrated dashboard. It is important to consider additional use cases that may apply to your specific network and what you may consider anomalous behavior.

- Detecting the absence of log generation is an event of high value. Such an event would indicate a service failure or even an intruder who has temporarily switched off logging or modified the log level to hide their tracks.

- Application events such as start and/or stop that were unscheduled would also be events to monitor and examine for possible security implications.

- OS events on the OpenStack service machines such as user logins, restarts also provide valuable insight into use/misuse

- Being able to detect the load on the OpenStack servers also enables responding by way of introducing additional servers for load balancing to ensure high availability.

- Other events that are actionable are networking bridges going down, ip tables being flushed on compute nodes and consequential loss of access to instances resulting in unhappy customers.

- To reduce security risks from orphan instances on a user/tenant/domain deletion in the Identity service there is discussion to generate notifications in the system and have

OpenStack components respond to these events as appropriate such as terminating instances, disconnecting attached volumes, reclaiming CPU and storage resources etc.

A cloud will host many virtual instances, and monitoring these instances goes beyond hardware monitoring and log files which may just contain CRUD events.

Security monitoring controls such as intrusion detection software, antivirus software, and spyware detection and removal utilities can generate logs that show when and how an attack or intrusion took place. Deploying these tools on the cloud machines provides value and protection. Cloud users, those running instances on the cloud may also want to run such tools on their instances.

参考資料

http://www.mirantis.com/blog/openstack-monitoring/

http://blog.sflow.com/2012/01/host-sflow-distributed-agent.html

http://blog.sflow.com/2009/09/lan-and-wan.html

http://blog.sflow.com/2013/01/rapidly-detecting-large-flows-sflow-vs.html

第46章 Case Studies: Monitoring and Logging

In this case study we discuss how Alice and Bob would address monitoring and logging in the public vs a private cloud. In both instances, time synchronization and a centralized store of logs become extremely important for performing proper assessments and troubleshooting of anomalies. Just collecting logs is not very useful, a robust monitoring system must be built to generate actionable events.

アリスのプライベートクラウド

In the private cloud, Alice has a better understanding of the tenants requirements and accordingly can add appropriate oversight and compliance on monitoring and logging. Alice should identify critical services and data and ensure that logging is turned at least on those services and is being aggregated to a central log server. She should start with simple and known use cases and implement correlation and alerting to limit the number of false positives. To implement correlation and alerting, she sends the log data to her organization's existing SIEM tool. Security monitoring should be an ongoing process and she should continue to define use cases and alerts as she has better understanding of the network traffic activity and usage over time.

ボブのパブリッククラウド

When it comes to logging, as a public cloud provider, Bob is interested in logging both for situational awareness as well as compliance. That is, compliance that Bob as a provider is subject to as well as his ability to provide timely and relevant logs or reports on the behalf of his customers for their compliance audits. With that in mind, Bob configures all of his instances, nodes, and infrastructure devices to perform time synchronization with an external, known good time device. Additionally, Bob's

team has built a Django based web applications for his customers to perform self-service log retrieval from Bob's SIEM tool. Bob also uses this SIEM tool along with a robust set of alerts and integration with his CMDB to provide operational awareness to both customers and cloud administrators.

第47章 コンプライアンス概要

OpenStackの環境構築において、監督当局からの要求、法的な要件、顧客ニーズ、プライバシーへの配慮、セキュリティのベストプラクティスなど、様々な理由でコンプライアンス活動が必要となるでしょう。コンプライアンス活動を適切に実施することで、このガイドで議論した他のセキュリティトピックスは統合、強化されます。この章の目的は以下の通りです。

- 共通のセキュリティ原則を確認する
- 業界認定や監督当局の認証を得るために必要な、共通コントロールフレームワークと認定リソースを説明する
- 監査人がOpenStack環境を評価する際のリファレンスとなる
- OpenStackおよびクラウド環境におけるプライバシーの考慮事項を説明する

セキュリティ原則

業界標準のセキュリティ原則は、コンプライアンス認証、認定のための基準を提供します。もしそれらの原則が対象のOpenStack環境で考慮、適用されていれば、認証を得る活動はシンプルになるでしょう。

1. 多層防御: クラウドアーキテクチャ内にあるリスクの存在場所を特定し、そのリスクを緩和すべく、コントロールします。特に懸念される部分では、多層防御はさらなるリスク緩和のため、相互補完的なコントロールを提供します。たとえば、クラウド内のテナント間の十分な独立性を確保するには、QEMUの強化、SELinuxサポートのハイパーバイザーを使う、強制アクセス制御の適用、攻撃対象面の縮小、などの対応を推奨します。この基本的な原則により、懸念される部分が強化されます。なぜなら仮に、ある階層が危険にさらされても、他の階層が防御し攻撃面を最小化するからです。

2. フェイルセーフ: 障害が発生した際に、システムは独立、安全な状態で停止するように構成されているべきです。たとえば、SSL証明書の検証では、もしそのCNAMEがサーバーのDNS名と一致しなければ、ネットワーク接続を切断し、停止すべきでしょう。CNAMEが一致しないのに接続の継続してしまうようなソフトウェアも存在します。それが安全性が低く、好ましくない状況であるにも関わらずです。

3. 最小権限: ユーザーとシステムサービスには最小限のアクセス権限のみを付与すべきです。アクセス権限は役割、責任と職務にもとづきます。この最小権限原則は、いくつかの国際セキュリティポリシーに明記されています。たとえば米国のNIST 800-53 AC-6項が挙げられます。

4. コンパートメント化: システムは、仮にあるマシンやシステムレベルのサービスが危険にさらされたとしても、影響がない他のシステムとは分離されているべきです。SELinuxの正しい使用は、この目標を達成するのに役立ちます。

5. プライバシー保護の奨励: システムとそのユーザーに関わる、収集可能な情報量は最小限とすべきです。

6. ロギング機能: 適切なロギングは、不正利用の監視や障害対応、証拠収集に役立ちます。多くの国において、それを再度証明する必要が無い、Common Criteria認定をうけた監査サブシステムの採用を強くおすすめします。

第48章 監査プロセスを理解する

情報システムのセキュリティコンプライアンスは、二つの基本的なプロセスの完了を前提としています。

1. セキュリティコントロールの実装と運用 情報システムを標準と規制の範囲内で運用しつづけること、それは、正式なアセスメント前でも行うべき内部活動です。なお監査人はこの時点で、ギャップ分析、助言、認証取得の可能性向上のために関与することがあります。

2. 独立した検査と検証 システムのセキュリティコントロールが標準と規制の範囲に従って実装され、効率的に運用されているか。これを中立的な第三者へ、認証を得る以前に証明しなければなりません。多くの認証はその継続を保証するため、包括的な継続監視の一部として、定期的な監査を必要とします。

監査の範囲を決める

何をコントロールするのか、OpenStack環境をいかにデザイン、変更していくかを明確にするため、監査範囲は初期の計画段階で決定すべきです。

OpenStack環境の範囲をコンプライアンス目的で明確化する際は、制御機能や仮想化技術など、慎重に扱うべきサービスの周辺を優先するよう、考慮すべきです。それらを妥協することは、OpenStack環境全体に影響を与えかねません。

範囲を限定することで、限定された環境に対し、OpenStackの設計者は高いセキュリティ品質を確立しやすくなります。しかしその取り組みの中で、セキュリティ強化の範囲や機能を不当に省かないことが重要です。典型的な例はPCI-DSSガイドラインです。決済に関わるインフラはセキュリティを精査されるでしょう。が、その影でその周辺サービスが放置されれば、そこが攻撃に対し無防備となります。

コンプライアンスに取り組む際、複数の認証で共通の領域と基準を明確にできれば、効率的に手間を減らすことができます。この本で取り上

げている監査原則とガイドラインの多くは、それらを特定するのに役立ちます。加えて、総合的なリストを提供するガイドラインが多くあります。以下に例を挙げます。

Cloud Security Alliance Cloud Controls Matrix (CCM)はクラウドプロバイダーのセキュリティを総合的に評価するにあたって、プロバイダーとユーザーの両方に役立ちます。CSA CCMはISO 27001/2、ISACA、COBIT、PIC、NIST、Jericho Forum、NERC CIPといった、多くの業界で認められた標準、規制をひも付けた統制フレームワークを提供します。

SCAP Security Guideはもうひとつの有用なリファレンスです。まだ出来たばかりですが、米国連邦政府の認証、推奨への対応に重点を絞ったツールとして普及すると予想されます。たとえば、SCAP Security Guideは現在、security technical implementation guides (STIGs)とNIST-800-53にある程度対応しています。

これらのコントロールマッピングは、認証間で共通の統制基準を特定します。また、監査人と被監査者両方にとって問題となる、特定のコンプライアンス認証、認定に必要なコントロールセットを可視化するのに役立ちます。

内部監査

クラウドが導入されたのであれば、内部監査が必要です。あなたが採用を決めた統制基準と、あなたのクラウドの設計、機能、配備戦略を比較する時です。目的はそれぞれの統制がどのように扱われているか、ギャップがどこに存在するか、理解することです。そして、その全てを将来のために文書化します。

OpenStackクラウドを監査するとき、OpenStackアーキテクチャー固有のマルチテナント環境を理解することが重要です。データの廃棄、ハイパーバイザーのセキュリティ、ノードの強化、および認証メカニズムなど、いくつか重要な部分があります。

外部監査に備える

内部監査の結果が良好であれば、いよいよ外部監査の準備です。この段階では、いくつかの鍵となる活動があります。概要は以下です。

- 内部監査での良好な状態を維持してください。それらは外部監査の実施期間に証明として役立ちます。またそれは、コンプライアンス統制に関する詳細な質疑応答の備えとなります。

- クラウドがコンプライアンスを維持し続けるために、自動テストツールを導入してください。
- 監査人を選ぶ

監査人の選定は困難を伴うことがあります。クラウドのコンプライアンス監査経験がある人を見つけてくるのが理想です。OpenStackの経験があれば、なお良しです。このプロセスを経験している人に相談するのがベストでしょう。なお、費用は契約の範囲と監査法人に大きく依存します。

外部監査

これが正式な監査プロセスです。監査人は、特定の認定向けのセキュリティ統制を確認し、これらの統制が監査期間において実行されていたか、その証明を求めます（たとえば、SOC 2監査は一般的に6-12ヶ月のセキュリティ統制を評価します）。どのような統制上の不具合も記録され、外部監査の最終報告書で文書化されます。OpenStack環境のタイプに依存しますが、これらの報告書を顧客はあとから見ることができます。それゆえ統制上の不具合を避けることは重要です。これが監査への準備が重要である理由です。

コンプライアンスの維持

このプロセスは一度の外部監査で終わることがありません。多くの認証は継続的なコンプライアンス活動、すなわち、定期的な監査を要求します。常に遵守を確実とするため、自動化されたコンプライアンス検証ツールをクラウド内に作ることをおすすめします。これは他のセキュリティ監視ツールに加えて実装されるべきです。このゴールがセキュリティおよびコンプライアンスであることを忘れないでください。これらのどちらかに不具合があれば、将来の監査で非常に面倒なことになります。

第49章 コンプライアンス活動

コンプライアンスのプロセスを大きく推進する、標準的な活動は数多くあります。この章ではいくつかの代表的なコンプライアンス活動を紹介します。これらはOpenStack固有ではありませんが、関係がわかるよう、このガイドの関連する節への参照も記載します。

Information Security Management System (ISMS)

Information Security Management System (ISMS)は包括的なポリシーとポロセスの集合です。組織が情報資産に関するリスクを管理するため、作成、維持します。もっとも一般的なクラウド向けISMSはISO/IEC 27001/2です。より厳格なコンプライアンス認証取得に向けて、セキュリティ統制と実践の確かな基盤を構築します。

リスク評価

リスク評価フレームワークは、組織やサービス内のリスクを特定します。また、それらのリスクと実装、緩和戦略それぞれの責任者を明確にします。リスクは全てのサービスで特定されるべきで、その範囲は技術統制から環境災害、人的要因など多岐にわたります。人的要因の例は、悪意ある内部監視者(や不良社員)などです。リスクは発生確率や影響度など、多様な指標を使って評価されます。OpenStack環境のリスク評価は、,このガイドで触れられている統制のギャップを含みます。

アクセスとログの検査

定期的なアクセスとログの検査は、認証、認可とサービス配備における責任を明確にするため、必要です。これらのトピックに関するOpenStack向けのガイダンスは、ロギングの節で詳細に説明します。

バックアップと災害対策

災害対策(Disaster Recovery, DR)とビジネス継続計画(Business Continuity Planning, BCP)はISMSとコンプライアンス活動で共通の要件です。それらの計画は定期的な検査と文書化が必要とします。OpenStackの主要領域はマネジメントセキュリティ領域にあたり、すべての単一障害点(Single Point of Failures, SPOFs)が特定されなければいけません。詳細は、安全なバックアップとリカバリーの節を参照してください。

セキュリティトレーニング

年次でのロール別セキュリティトレーニングは、ほぼすべてのコンプライアンス認証、認定で必須の要件です。セキュリティトレーニングの効果を最適化するため、一般的にはロール別に実施します。たとえば開発者、運用担当者、非技術者別、などです。加えて、このガイドにもとづくクラウド、OpenStackセキュリティに関するトレーニングの実施が理想的でしょう。

セキュリティの検査

OpenStackは人気のあるオープンソースプロジェクトです。多くのソースコードとアーキテクチャーはデベロッパー、組織、企業によって精査されています。これはセキュリティの観点から大きな利点ですが、セキュリティ検査はサービスプロバイダーにとって、それでもなお重大な懸念事項です。環境は変化しつづけますが、セキュリティは必ずしも開発者の一番の関心事ではないからです。包括的なセキュリティ検査プロセスとして、アーキテクチャー検査、脅威のモデリング、ソースコード分析と侵入テストなどが挙げられます。そして、セキュリティ検査には広く公開されている多くのテクニックと推奨があります。よくテストされた例として、Microsoft Trustworthy Computing Initiativeのとりくみとして作成された、Microsoft SDLがあります。

脆弱性の管理

セキュリティアップデートはプライベート、パブリックを問わず、あらゆるIaaS環境において重要です。脆弱なシステムは攻撃面を広げ、攻撃者にターゲットをさらしてしまいます。一般的なスキャニング技術と脆弱性検知サービスはこの脅威を和らげるのに役立ちます。スキャンが認証されたものであり、その緩和戦略が単なる境界線の防御力向上にとどまらないことが重要です。OpenStackのようなマルチテナントアーキテク

チャーは特にハイパーバイザーの脆弱性に影響されやすく、それはシステムの脆弱性管理の重点項目です。詳細はインスタンス隔離の節を参照してください。

データの分類

データの分類作業は、多くの場合、顧客情報を事故、故意の窃盗、損失、不適切な公開から保護するため、情報の分類と扱いの方法を定義します。一般的にこの作業は、情報を機密性の有無、個人識別の可不可(Personally Identifiable Information, PII)による分類を含みます。使用される基準はその環境、背景によって様々です(政府、ヘルスケアなど)。そして根本的な原則は、そのデータ分類が明確に定義され、通常利用されていることです。もっとも一般的な保護メカニズムには、業界標準の暗号化技術が挙げられます。詳細はデータセキュリティの節を参照してください。

例外プロセス

例外プロセスはISMSの重要な要素です。とある行動が組織の定義したセキュリティポリシーに準拠していない場合、それは記録されなければいけません。適正な理由と緩和策の詳細が含まれ、関係当局に認められる必要があります。OpenStackのデフォルト構成は、様々なコンプライアンス基準、記録されるべきコンプライアンス基準を満たすべく、変化していくでしょう。またそれは、コミュニティへの貢献によって修正されていく可能性があります。

第50章 認証とコンプライアンスの報告書

コンプライアンスとセキュリティは排他的でなく、あわせて取り組むべきものです。OpenStack環境は、セキュリティの強化なしに、コンプライアンス要件を充足することができないでしょう。以下のリストは、OpenStackアーキテクト向けの、商業規格および政府機関の認証を得るための基本的な知識とガイダンスです。

商業規格

OpenStackの商用環境向けには、まずは開始点として、SOC 1/2とISO 27001/2の検討を推奨します。そこで要求されるセキュリティ活動を確実に実行することで、セキュリティのベストプラクティスと共通統制基準を導入を促進し、政府系認定などの、より厳格なコンプライアンス活動の取得にも役立ちます。

これらの基本的認証を取得したのち、より環境特有の認証を検討します。たとえば、クラウドがクレジットカードのトランザクションを扱うのであればPCI-DSSが必要ですし、ヘルスケア情報を保持するならHIPPAが、連邦政府向けにはFedRAMP/FISMA、ITAR認証が必要となるでしょう。

SOC 1 (SSAE 16) / ISAE 3402

Service Organization Controls (SOC)基準は米国公認会計士協会 - American Institute of Certified Public Accountants (AICPA)によって定められています。SOC統制はサービスプロバイダーの関連財務諸表と主張を評価します。たとえばSarbanes-Oxley法への準拠などです。SOC 1はStatement on Auditing Standards No. 70 (SAS 70) Type II 報告書を代替します。これらの統制は物理的なデータセンターを評価範囲に含みます。

SOC 1報告書には二つの種類があります。

- Type 1 - report on the fairness of the presentation of management's description of the service organization's system

and the suitability of the design of the controls to achieve the related control objectives included in the description as of a specified date.

- Type 2 - report on the fairness of the presentation of management's description of the service organization's system and the suitability of the design and operating effectiveness of the controls to achieve the related control objectives included in the description throughout a specified period

詳細はAICPA Report on Controls at a Service Organization Relevant to User Entities' Internal Control over Financial Reportingを参照してください。

SOC 2

Service Organization Controls (SOC) 2は、サービス提供組織がユーザーデータとその情報の機密性とプライバシーを制御するために使っているシステムのセキュリティ、可用性、および処理の完全性に関する統制の自己証明です。ユーザーの例は、サービス組織を統制する人、サービス組織の顧客、監視当局、ビジネスパートナー、サプライヤー、およびサービス組織の理解者やそれを統制する人です。

SOC 2報告書には二つの種類があります。

- Type 1 - report on the fairness of the presentation of management's description of the service organization's system and the suitability of the design of the controls to achieve the related control objectives included in the description as of a specified date.

- Type 2 - report on the fairness of the presentation of management's description of the service organization's system and the suitability of the design and operating effectiveness of the controls to achieve the related control objectives included in the description throughout a specified period.

詳細はAICPA Report on Controls at a Service Organization Relevant to Security, Availability, Processing Integrity, Confidentiality or Privacyを参照してください。

SOC 3

Service Organization Controls (SOC) 3はサービス提供組織のための公的なサービス報告書です。これらのレポートはサービス組織のセ

キュリティ、可用性、処理の完全性、機密性、またはプライバシーに関する統制の保証を求めるユーザーニーズを満たすためのレポートです。ただし、SOC 2報告書ほどの情報は必要ありません。SOC 3報告書はAICPA/Canadian Institute of Chartered Accountants (CICA)のTrust Services Principles, Criteria, and Illustrations for Security, Availability, Processing Integrity, Confidentiality, and Privacyをもって作成されています。SOC 3は一般的に使われる報告書であり、Webサイト上で証明書として自由に配布できます。

詳細はAICPA Trust Services Report for Service Organizationsを参照してください。

ISO 27001/2

ISO/IEC 27001/2はBS7799-2の後継標準で、Information Security Management System (ISMS)の要件です。ISMSは組織が情報資産のリスクを管理するために作成、維持する、ポリシーとプロセスの包括的なセットです。それらのリスクはユーザー情報のConfidentiality - 機密性、Integrity - 完全性、および Availability - 可用性 (CIA)に深く関係しています。CIAセキュリティの三要素は、このガイドの多くの章で基本となっています。

詳細はISO 27001を参照してください。

HIPAA / HITECH

Health Insurance Portability and Accountability Act (HIPAA)は米国の健康保険における可搬性と責任に関する法律で、カルテ情報の収集、保存、および廃棄に関するルールを定めています。この法律は、保護医療情報 (Protected Health Information, PHI)は、権限のない人が”利用できない、読めない、複合できない”ように変換されなければいけないこと、また、データが保存中でも、処理中でも、暗号化するべきであることに言及しています。

HIPAA is not a certification, rather a guide for protecting healthcare data. Similar to the PCI-DSS, the most important issues with both PCI and HIPPA is that a breach of credit card information, and health data, does not occur. In the instance of a breach the cloud provider will be scrutinized for compliance with PCI and HIPPA controls. If proven compliant, the provider can be expected to immediately implement remedial controls, breach notification responsibilities, and significant expenditure on additional compliance activities. If not compliant, the cloud

provider can expect on-site audit teams, fines, potential loss of merchant ID (PCI), and massive reputation impact.

Users or organizations that possess PHI must support HIPAA requirements and are HIPAA covered entities. If an entity intends to use a service, or in this case, an OpenStack cloud that might use, store or have access to that PHI, then a Business Associate Agreement must be signed. The BAA is a contract between the HIPAA covered entity and the OpenStack service provider that requires the provider to handle that PHI in accordance with HIPAA requirements. If the service provider does not handle the PHI, such as with security controls and hardening, then they are subject to HIPAA fines and penalties.

OpenStackアーキテクトはHIPAAの条項を解釈し、対応します。データ暗号化はその中核となる活動です。現在、OpenStack環境に保存される、いかなる保護カルテ情報にも暗号化を要求され、業界標準の暗号化アルゴリズムの採用が期待されます。なお、将来予定されている、たとえばオブジェクト暗号化などのOpenStackプロジェクトは、法令遵守のためHPAAガイドラインの適用を促進するでしょう。

詳細はHealth Insurance Portability And Accountability Actを参照してください。

PCI-DSS

Payment Card Industry Data Security Standard (PCI DSS)はPayment Card Industry Standards Councilで定義されました。目的は、クレジットカード不正の防止のため、カード所有者情報に関する統制度を向上することです。コンプライアンス検査は年次で、外部のコンプライアンス評価報告書(Report on Compliance, ROC)を作成する認定評価機関 (Qualified Security Assessor, QSA)、もしくは、自己評価問診票(Self-Assessment Questionnaire, SAQ)によって実施されます。これはカード所有者のトランザクション量に依存します。

カード情報を保存、処理、転送するOpenStack環境は、PCI-DSSの対象です。カード情報を扱うシステムやネットワークが正しく分離されていないすべてのOpenStackコンポーネントは、PCI-DSSのガイドラインに適合しません。PCI-DSSでいう分離は、マルチ手ナンシーを認めておらず、(サーバーおよびネットワークの)物理的な分離が必要です。

詳細はPCI security standardsを参照してください。

政府標準

FedRAMP

”Federal Risk and Authorization Management Program (FedRAMP)は米国連邦政府全体のプログラムであり、クラウド製品とサービスのセキュリティ評価、認証、および継続的モニタリングの、標準化された手順を提供します” NIST 800-53はFISMAとRedRAMPの両方の基礎であり、特にクラウド環境における保護を提供するために選択されたセキュリティ統制を強制します。セキュリティ統制に関する具体性と政府標準を満たすための文書量を、FedRAMPは徹底しています。

詳細はhttp://www.gsa.gov/portal/category/102371を参照してください。

ITAR

International Traffic in Arms Regulations (ITAR)は米国政府規制の集合であり、米国軍需品リスト(United States Munitions List, USML)と関連技術情報に関係する防衛物品・サービスの輸出入を統制します。ITARは正式な認証というより、”軍事活動支援”の位置づけでクラウドプロバイダーから提示されます。この統制は一般的に、NIST 800-53フレームワークにもとづき、分離されたクラウド環境の実装を意味します。FISMA要件により、米国民かつ身元審査された人のみがアクセスできるよう、追加の統制で補完します。

詳細はhttp://pmddtc.state.gov/regulations_laws/itar_official.htmlを参照してください。

FISMA

米国連邦情報セキュリティマネジメント法 - Federal Information Security Management Act requires、FISMAは、政府機関は多数の政府セキュリティ標準を実装するために、包括的な計画を作成する必要があるとして、2002年 電子政府法 - E-Government Act of 2002 内で制定されました。FISMAは多数のNIST公表文献を活用し、政府のデータを保存、処理する情報システムを作成するためのプロセスを説明しています。

このプロセスは三つの主要カテゴリに分割されています。

- システムのカテゴリ分け 情報システムは連邦情報処理規格(Federal Information Processing Standards Publication 199, FIPS 199)で定

められたセキュリティカテゴリに分類されます。これらのカテゴリはシステムの情報漏洩の潜在的な影響を反映しています。

- 統制の選択 FIPS 199で定められたシステムセキュリティのカテゴリにもとづき、組織は情報システムのための特定のセキュリティ統制要求を特定すべく、FIPS 200を活用します 。たとえば、もしシステムが”中程度”と分類されているのであれば、安全なパスワードの強制が求められるでしょう。

- Control TailoringOnce system security controls are identified, an OpenStack architect will utilize NIST 800-53 to extract tailored control selection. For example, specification of what constitutes a “secure password.”

第51章 プライバシー

プライバシーはコンプライアンスプログラムの重要な要素になりつつあります。顧客はプライバシーの観点から、データがいかに扱われているか関心を高めており、データを扱う企業はより高い基準を期待されています。

OpenStack環境では、組織のプライバシーポリシー、米国 - EU間のセーフハーバーフレームワーク、ISO/IEC 29100:2011 プライバシーフレームワークなど、プライバシー特化ガイドライン遵守の証明を求められることが多いです。米国ではAICPAが重視すべき10のプライバシー項目を公表しており、ビジネス用途のOpenStack環境はそのうちのいくつか、もしくは全原則の立証を期待されます。

個人情報の保護に取り組むOpenStackアーキテクトを支援するため、OpenStackアーキテクトには、NIST刊行 800-122 "Guide to Protecting the Confidentiality of Personally Identifiable Information (PII).をおすすめします。このガイドは以下を保護するプロセスについて述べています。

> "政府機関が保有するあらゆる個人情報、(1)個人を特定、追跡しうるあらゆる情報、たとえば氏名、社会保障番号、出生年月日、出生地、母の旧姓、生体情報など。および、(2)個人に結びつく、結びつけられるあらゆる情報、たとえば医療、教育、金融、雇用情報など"

包括的なプライバシー管理には、十分な準備、考慮と投資が必要です。また、グローバルなOpenStackクラウドの構築時には、さらなる複雑さに気づくでしょう。米国および、それより厳しいEUのプライバシー法令の違いが良い例です。加えて、クレジットカード番号や医療情報など、機密性の高い個人情報を扱う場合にはさらなる注意が必要です。これら機密性の高い情報はプライバシー法令だけでなく、監視当局や政府規制にも関連します。政府によって発行されたものなど、ベストプラクティスに従うことで、OpenStack環境向けの総合的なプライバシー管理ポリシーが確立、実践されていくでしょう。

第52章 ケーススタディ: コンプライアンス

このケーススタディでは、アリスとボブがどのように一般的なコンプライアンス要件に対応するかを説明します。これまでの章で、さまざまなコンプライアンス認証と標準について言及しました。アリスはプライベートクラウドでコンプライアンスに取り組み、いっぽうボブはパブリッククラウド向けのコンプライアンスに注力します。

アリスのプライベートクラウド

アリスはOpenStackプライベートクラウドを米国政府向けに構築しています。具体的には、信号処理向けの柔軟なコンピューティング環境です。アリスは政府向けコンプライアンス要件を調査した結果、これから構築しようとしているプライベートクラウドはFISMAおよびFedRAMP認定が必要であると判断しました。これは政府系機関、行政部、および契約者、どのような立場であっても、認定クラウドプロバイダー(Certified Cloud Provider, CCP)になるために必要な手続きです。特に信号処理は、FISMAはそれを”深刻で壊滅的な影響”をシステムに与えうるとしているため、FISMA影響度が”高”となりがちです。加えてFISMA Moderateレベルにおいて、アリスはそのプライベートクラウドを確実にFedRAMP認証としなければいけません。これはクラウド内に政府の情報を保有する、全ての機関に求められてる条件です。

これらの厳しい政府規制の要件を満たすため、アリスは多くの活動を行います。範囲の決定作業は、実装すべき統制の量に影響するため、特に重要です。これはNIST刊行 800-53で定められています。

All technology within her private cloud must be FIPS certified technology, as mandated within NIST 800-53 and FedRAMP. As the U.S. Department of Defense is involved, Security Technical Implementation Guides (STIGs) will come into play, which are the configuration standards for DOD IA and IA-enabled devices / systems. Alice notices a number of complications here as there is no STIG for OpenStack, so she must address several underlying requirements for each OpenStack service; for example, the networking SRG and Application SRG will both be applicable (list of SRGs). Other critical controls include ensuring that all

identities in the cloud use PKI, that SELinux is enabled, that encryption exists for all wire-level communications, and that continuous monitoring is in place and clearly documented. Alice is not concerned with object encryption, as this will be the tenants responsibility rather than the provider.

もしアリスが十分な範囲を定義し、それらのコンプライアンス活動を実施できたのであれば、次は認定外部監査人によるFedRAMP認証の取得プロセスに移ります。一般的にこのプロセスは最長6ヶ月を要します。このステップを経て、Authority to Operate - 注意影響レベル認定 を取得し、OpenStackクラウドサービスを政府に提案できるようになります。

ボブのパブリッククラウド

ボブは新たなOpenStackクラウド環境のコンプライアンス活動を任されています。このクラウドは小規模の開発者やスタートアップだけでなく、大規模企業向けにも注力しています。ボブは個人開発者はコンプライアンス認証を意識することが多くないが、いっぽうで大規模企業向けには認証が重要であることを認識しています。ボブは特にSOC 1、SOC 2、およびISO 27001/2認証を早急に取得したいと考えています。そこでボブは3つの認証に共通する統制を特定するため、Cloud Security Alliance Cloud Control Matrix (CCM)を参考にしました（たとえば、定期的なアクセス検査、監査可能なロギングや監視サービス、リスク評価活動、セキュリティレビューなど)。それからボブは、パブリッククラウドのギャップ評価、結果のレビュー、そして特定されたギャップを埋めるため、経験ある監査人チームと契約します。ボブは他のチームメンバーとともに、それらのセキュリティ統制と活動が一般的な監査期間(～6-12ヶ月)において、定期的に、確実に機能するようにします。

監査期間の最後にボブは外部監査人チームとの調整を行います。目的は、6ヶ月以上にわたって無作為なタイミングで実施した、セキュリティ統制のレビューです。そして、監査人チームはボブにSOC 1とSOC 2、また別途ISO 27001/2向けの公式な報告書を提供します。ボブのパブリッククラウド採用における勤勉な取り組みの結果、指摘されるような追加のギャップはありませんでした。ボブは正式な報告書を彼の顧客にNDA下で提供でき、また、SOC 1、SOC 2、およびISO 27001/2に準拠していることを彼のウェブサイトでアピールできるようになりました。

付録A コミュニティのサポート

目次

OpenStackの利用に役立つ、多くのリソースがあります。OpenStackコミュニティのメンバーはあなたの質問に回答するでしょうし、バグ調査のお手伝いもします。コミュニティはOpenStackを継続的に改善、機能追加していますが、もしあなたが何らかの問題に直面したら、遠慮せずに相談してください。下記のリソースをOpenStackのサポートとトラブルシュートに活用して下さい。

ドキュメント

OpenStackのドキュメントは、 docs.openstack.orgを参照してください。

ドキュメントにフィードバックするには、 OpenStack Documentation Mailing Listの <openstack-docs@lists.openstack.org>か、Launchpadのreport a bugを活用してください。

OpenStackクラウドと関連コンポーネントの導入ガイド:

- Installation Guide for Debian 7.0
- Installation Guide for openSUSE and SUSE Linux Enterprise Server
- Red Hat Enterprise Linux, CentOS, and Fedora向けインストールガイド
- Ubuntu 12.04 (LTS)向けインストールガイド

OpenStackクラウドの構成と実行ガイド:

- Cloud Administrator Guide
- Configuration Reference
- Operations Guide
- High Availability Guide
- Security Guide
- Virtual Machine Image Guide

OpenStackダッシュボードとCLIクライアントガイド

- API Quick Start
- End User Guide
- Admin User Guide
- コマンドラインインターフェースのリファレンス

OpenStack APIのリファレンスガイド

- OpenStack API Reference
- OpenStack Block Storage Service API v2 Reference
- OpenStack Compute API v2 and Extensions Reference
- OpenStack Identity Service API v2.0 Reference
- OpenStack Identity Service API v2.0 Reference
- OpenStack Networking API v2.0 Reference
- OpenStack Object Storage API v1 Reference

トレーニングガイドはクラウド管理者向けのソフトウェアトレーニングを提供します。

ask.openstack.org

OpenStackの導入やテスト中、特定のタスクが完了したのか、うまく動いていないのかを質問したくなるかもしれません。その時

は、ask.openstack.orgが役に立ちます。ask.openstack.orgで、すでに同様の質問に回答がないかを確かめてみてください。もしなければ、質問しましょう。簡潔で明瞭なサマリーをタイトルにし、できるだけ詳細な情報を記入してください。コマンドの出力結果やスタックトレース、スクリーンショットへのリンクなどがいいでしょう。

OpenStack メーリングリスト

回答やヒントを得るとっておきの方法は、OpenStackメーリングリストへ質問や問題の状況を投稿することです。同様の問題に対処したことのある仲間が助けてくれることでしょう。購読の手続き、アーカイブの参照はhttp://lists.openstack.org/cgi-bin/mailman/listinfo/openstackで行ってください。特定プロジェクトや環境についてのメーリングリストは、on the wikiで探してみましょう。すべてのメーリングリストは、http://wiki.openstack.org/MailingListsで参照できます。

OpenStack wiki

OpenStack wikiは広い範囲のトピックを扱っていますが、情報によっては、探すのが難しかったり、情報が少なかったりします。幸いなことに、wikiの検索機能にて、タイトルと内容で探せます。もし特定の情報、たとえばネットワークや novaについて探すのであれば、多くの関連情報を見つけられます。日々追加されているため、こまめに確認してみてください。OpenStack wikiページの右上に、その検索窓はあります。

Launchpad バグエリア

OpenStackコミュニティはあなたのセットアップ、テストの取り組みに価値を感じており、フィードバックを求めています。バグを登録するには、https://launchpad.net/+loginでLaunchpadのアカウントを作成してください。Launchpadバグエリアにて、既知のバグの確認と報告ができます。すでにそのバグが報告、解決されていないかを判断するため、検索機能を活用してください。もしそのバグが報告されていなければ、バグレポートを入力しましょう。

使いこなすヒント:

- 明瞭で簡潔なまとめを!
- できるだけ詳細な情報を記入してください。コマンドの出力結果やスタックトレース、スクリーンショットへのリンクなどがいいでしょう。

- ソフトウェアとパッケージのバージョンを含めることを忘れずに。特に開発ブランチは"Grizzly release" vs git commit bc79c3ecc55929bac585d04a03475b72e06a3208のように明記しましょう。
- 環境固有のお役立ち情報、たとえばUbuntu 12.04や複数ノードインストール

Launchpadバグエリアは下記リンクを参照してください。

- Bugs: OpenStack Block Storage (cinder)
- Bugs: OpenStack Compute (nova)
- Bugs : OpenStack Dashboard (horizon)
- Bugs : OpenStack Identity (keystone)
- Bugs : OpenStack Image Service (glance)
- Bugs : OpenStack Networking (neutron)
- Bugs : OpenStack Object Storage (swift)
- Bugs: Bare Metal (ironic)
- Bugs: Data Processing Service (sahara)
- Bugs: Database Service (trove)
- Bugs: Orchestration (heat)
- Bugs: Telemetry (ceilometer)
- Bugs: Queue Service (marconi)
- Bugs: OpenStack API Documentation (api.openstack.org)
- Bugs: OpenStack Documentation (docs.openstack.org)

OpenStack IRC チャネル

OpenStackコミュニティはFreenode上の#openstack IRCチャネルを活用しています。あなたはそこに訪れ、質問することで、差し迫った問題へのフィードバックを迅速に得られます。IRCクライアントをインストール、もしくはブラウザベースのクライアントを使う

には、http://webchat.freenode.net/にアクセスしてください。また、Colloquy (Mac OS X, http://colloquy.info/), mIRC (Windows, http://www.mirc.com/), or XChat (Linux)なども使えます。IRCチャネル上でコードやコマンド出力結果を共有したい時には、Paste Binが多く使われています。OpenStackプロジェクトの Paste Binはhttp://paste.openstack.orgです。長めのテキストやログであっても、webフォームに貼り付けてURLを得るだけです。OpenStack IRCチャネルは、#openstack on irc.freenode.netです。OpenStack関連IRCチャネルは、https://wiki.openstack.org/wiki/IRCにリストがあります。

ドキュメントへのフィードバック

ドキュメントにフィードバックするには、 OpenStack Documentation Mailing Listの <openstack-docs@lists.openstack.org>か、Launchpadのreport a bugを活用してください。

OpenStackディストリビューション

OpenStackのコミュニティサポート版を提供しているディストリビューション

- Debian: http://wiki.debian.org/OpenStack
- CentOS、Fedora、およびRed Hat Enterprise Linux: http://openstack.redhat.com/
- openSUSEとSUSE Linux Enterprise Server: http://en.opensuse.org/Portal:OpenStack
- Ubuntu: https://wiki.ubuntu.com/ServerTeam/CloudArchive

用語集

ACL
: アクセス制御リスト参照。

AMQP
: Advanced Message Queue Protocol. An open Internet protocol for reliably sending and receiving messages. It enables building a diverse, coherent messaging ecosystem.

API
: アプリケーションプログラミングインターフェース。

BMC
: Baseboard Management Controller. The intelligence in the IPMI architecture, which is a specialized micro-controller that is embedded on the motherboard of a computer and acts as a server. Manages the interface between system management software and platform hardware.

CA
: Certificate Authority or Certification Authority. In cryptography, an entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate. This enables others (relying parties) to rely upon signatures or assertions made by the private key that corresponds to the certified public key. In this model of trust relationships, a CA is a trusted third party for both the subject (owner) of the certificate and the party relying upon the certificate. CAs are characteristic of many public key infrastructure (PKI) schemes.

Chef
: An operating system configuration management tool supporting OpenStack deployments.

CMDB
: 構成管理データベース。

DAC
: Discretionary access control. Governs the ability of subjects to access objects, while enabling users to make policy decisions and assign security attributes. The traditional UNIX system of users, groups, and read-write-execute permissions is an example of DAC.

DHCP
: Dynamic Host Configuration Protocol. A network protocol that configures devices that are connected to a network so they can communicate on that network by using the Internet Protocol (IP). The protocol is implemented in a client-server model where DHCP clients request configuration data such as, an IP address, a default route, and one or more DNS server addresses from a DHCP server.

Django
: Horizon 中で広く使用される Web フレームワーク。

DNS
: Domain Name Server. A hierarchical and distributed naming system for computers, services, and resources connected to the Internet or a private network. Associates a human-friendly names to IP addresses.

Puppet
: An operating system configuration management tool supported by OpenStack.

Qpid
: Message queue software supported by OpenStack, an alternative to RabbitMQ.

RabbitMQ
: The default message queue software used by OpenStack.

SPICE
: The Simple Protocol for Independent Computing Environments (SPICE) provides remote desktop access to guest virtual machines. It is an alternative to VNC. SPICE is supported by OpenStack.

Virtual Network Computing (VNC)
: Open source GUI and CLI tools used for remote console access to VMs. Supported by Compute.

ZeroMQ
: Message queue software supported by OpenStack. An alternative to RabbitMQ. Also spelled 0MQ.