OpenStack セキュリティガイド

[FAMILY Given]

havana (2014-03-03)

概要

本書は OpenStack クラウドを安全にするためのベストプラクティスと基本的な考え方について書かれています。

目次

図の一覧

はじめに

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	• Object Storage に関する章を追加しました。
October 17, 2013	• Havana リリース。
July 2, 2013	• 初版作成...

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 の導入の増加と製品の成熟と伴い、セキュリティへ対する要件が高くなり、OpenStack Security Group で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。 さまざまな環境で OpenStack の導入やセキュア化の専門知識を持つ著者が豊富なノウハウを本書にて共有します。

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 環境のセキュリティを強化したり、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の参考書として活用してください。

本書の目的

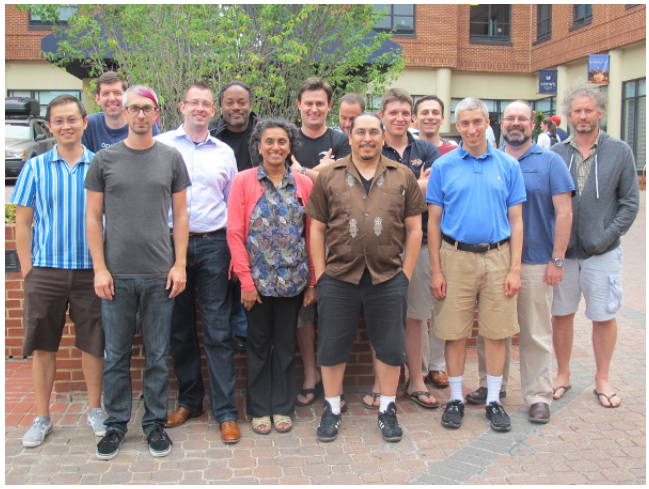
- OpenStack のセキュリティ領域の明確化
- OpenStack をセキュア化するガイドの提供
- 現在の OpenStack におけるセキュリティ懸念事項と実現可能な軽減策の紹介
- 今後予定されているセキュリティ機能の議論
- コミュニティ主導のナレッジ蓄積と普及の場の提供

本書の執筆方法

本書はOpenStack Operations Guide（OpenStack 運用ガイド）と同様に「Book Sprint メソッド」を用いました。このメソッドでは、迅速な大量文章の作成を実現します。OpenStack Security Group のコーディネーターは再びAdam Hydeをファシリテーターとして力を借りました。さら

に企業からのサポートが得られ、オレゴン州ポートランドで開催された OpenStack サミットでプロジェクトが正式に公表されました。

グループの主要なメンバーが集まるために、執筆チームはメリーランド州アナポリスに集まりました。この集まりは、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。



チームメンバー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 は VMware の Networking and Security Business Unit の Senior Solution Architect です。顧客が OpenStack や VMware NSX（以前は Nicira の Network Virtualization Platform として知られていました）を導入する手助けをしています。VMware（Nicira の企業買収により）に参加する前は、Q1 Labs、Symantec、Vontu、Brightmail に勤務していました。U.C. Berkeley の Electrical Engineering/Computer Science、Nuclear Engineering の B.S. を保持してます。また、University of San Francisco の MBA を保持しています。

- 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。

本書は5日間の Book Sprint で作成されました。Book Sprintで は、3-5日でドキュメントを作成するために、高度なコラボレーションと統制されたプロセスによってグループメンバーをひとつにします。Book SprintメソッドはAdam Hyde によって設立された高度なファシリテーションプロセスです。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。

初版の発行後、以下の内容を追加しました。

- Rodney D. Beede, Seagate Technology

  Rodney D. Beede は Seagate Technology の Cloud Security Engineer です。彼は OpenStack Object Storage (Swift) のセキュア化に関する不足していた章に貢献しました。University of Colorado の Computer Science に関する M.S. を保持しています。

このドキュメントへの貢献方法

執筆作業の初めは空調が効きすぎの部屋で行われました。最終的に、その部屋がグループのオフィスとして執筆スプリント期間中使用されました。

OpenStack ドキュメントに貢献する方法について: 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

パブリッククラウドの対極にあるのがプライベートクラウドです。NIST は、プライベートクラウドを、事業組織などの複数の利用者から成る単一の組織の専用使用のために提供されるクラウドと定義しています。プライベートクラウドの所有、管理、および運用は、その組織、第三者、もしくはそれらの組み合わせにより行われ、存在場所としては、その組織の施設内または外部の場合があります。プライベートクラウドのユースケースは多様であるため、セキュリティ課題もそれぞれで異なります。

コミュニティクラウド

NIST では、コミュニティクラウドを、共通の関心事（例えば、任務、セキュリティの必要、ポリシー、法令順守に関わる考慮事項 ）を持つ複数の組織から成る特定の利用者の共同体の専用使用のために提供されるクラウドと定義しています。コミュニティクラウドの所有、管理、および運用は、共同体内の 1 つまたは複数の組織、第三者、もしくはそれらの組み合わせにより行われ、存在場所はその組織の施設内または外部の場合があります。

ハイブリッドクラウド

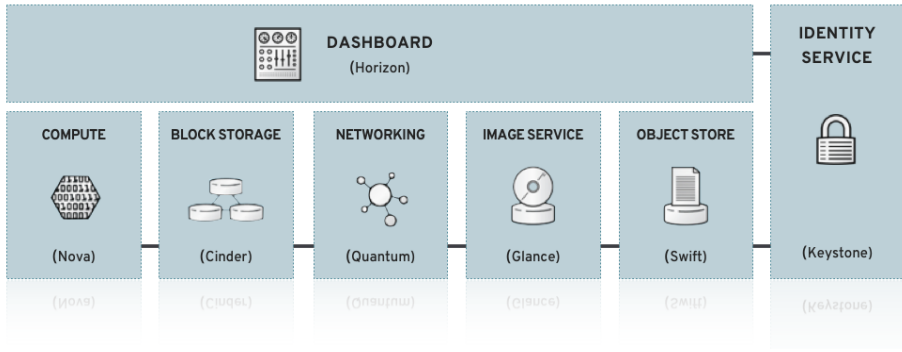
NIST では、ハイブリッドクラウドを、2 つ以上の異なるクラウドインフラストラクチャー（プライベート、コミュニティ、パブリック）を組み合わせたクラウドと定義しています。各クラウドは、依然として独自のエンティティですが、データおよびアプリケーションの移植性を可能にする標準化された技術あるいは専有技術（例: クラウド間のロードバランスのためのクラウドバーストなど）により結合されます。例えば、オ

ンライン小売業者は、柔軟なプロビジョニングが可能なパブリッククラウドに広告やカタログを掲示している場合があります。これにより、柔軟かつ費用対効果の高い方法で季節的な負荷に対応することが可能となります。顧客が発注処理を開始すると、よりセキュアなプライベートクラウドのバックエンドに転送されます。

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) は、多層アプリケーション、開発/テスト環境、「ビッグデータ」を処理する Hadoop のクラスター、ハイパフォーマンスコンピューティングなどをホストする、大規模な仮想マシンインスタンスの管理をサポートするサービスを提供します。

Compute Service は、サポート対象のハイパーバイザーと連動する抽象化レイヤーを介してこのような管理を行います。ハイパーバイザーについては、後半で詳しく説明します。

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。

OpenStack のデプロイメントでは、Compute のセキュリティが極めて重要となります。セキュリティ強化のテクニックには、頑強なインスタンスの隔離、Compute のサブコンポーネント間におけるセキュアな通信、一般向けの API エンドポイントの回復性などがあげられます。

オブジェクトストレージ

OpenStack Object Storage Service (Swift) は、クラウド内の任意データの保管/取得機能のサポートを提供します。Object Storage Service はネイティブ API および Amazon Web Services S3 互換の API の両方を提供します。このサービスは、データレプリケーションにより高度な回復性を提供し、ペタバイト規模のデータの処理が可能です。

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

OpenStack Block Storage Service (Cinder) は、Compute インスタンス用に永続的なブロックストレージを提供します。Block Storage Service はブロックデバイスの作成からインスタンスへのボリュームの接続、それらの解放にいたるまでのライフサイクルを管理する役割を果たします。

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

OpenStack Networking Service (Neutron、旧称 Quantum) はIP アドレス管理、DNS、DHCP、負荷分散、セキュリティグループ (ファイアウォールのポリシーなど、ネットワークのアクセスルール) など、さまざまなネットワークサービスをクラウドユーザー (テナント) に提供します。また、各種ネットワークソリューションとのプラグ可能な統合を可能に

するソフトウェア定義ネットワーク(SDN) のフレームワークを提供します。

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。

Dashboard

OpenStack Dashboard Service (Horizon) は、クラウド管理者とクラウドテナントの両方に向けた Web ベースのインターフェースを提供します。このインターフェースにより、管理者およびテナントは、クラウドリソースのプロビジョニング、管理、監視を行うことができます。Horizon は通常、一般i向けにデプロイされ、パブリック Web ポータルの一般的なセキュリティ問題が伴います。

Identity サービス

OpenStack Identity Service (Keystone) は、クラウドインフラストラクチャー全体にわたる認証および承認サービスを提供する共有サービスです。Identity Service には、複数形式の認証に対するプラグ可能なサポートを採用しています。

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

OpenStack Image Service (Glance) は、ディスクイメージ管理サービスを提供します。Image Service は、必要に応じて、イメージの検索、登録、デリバリサービスを Compute サービスである Compute に提供します。

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

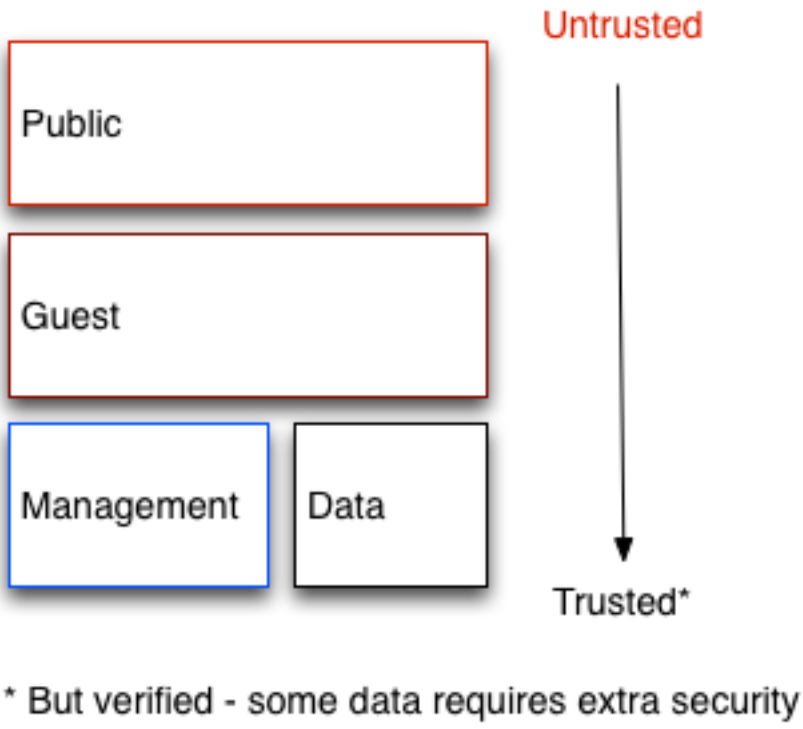
2. ゲスト

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

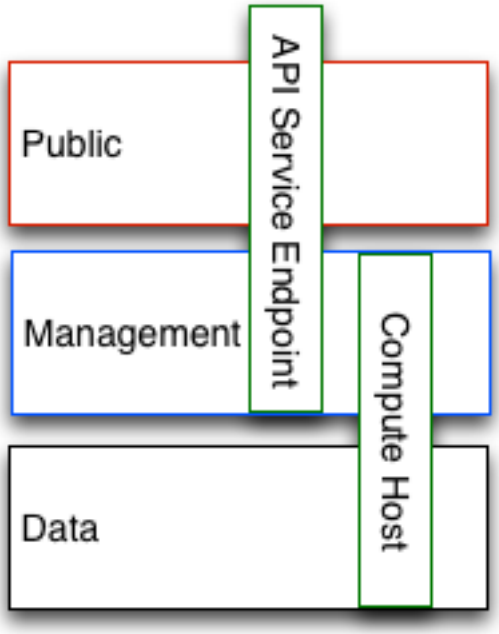
データセキュリティドメインは主に、OpenStack ではストレージサービスの情報に関係します。このネットワークを通過するデータの多くは、完全性や機密性に関する強力な要件を持ち、デプロイメントの種類によっては可用性に関する強力な要件が出てくる場合があります。

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。

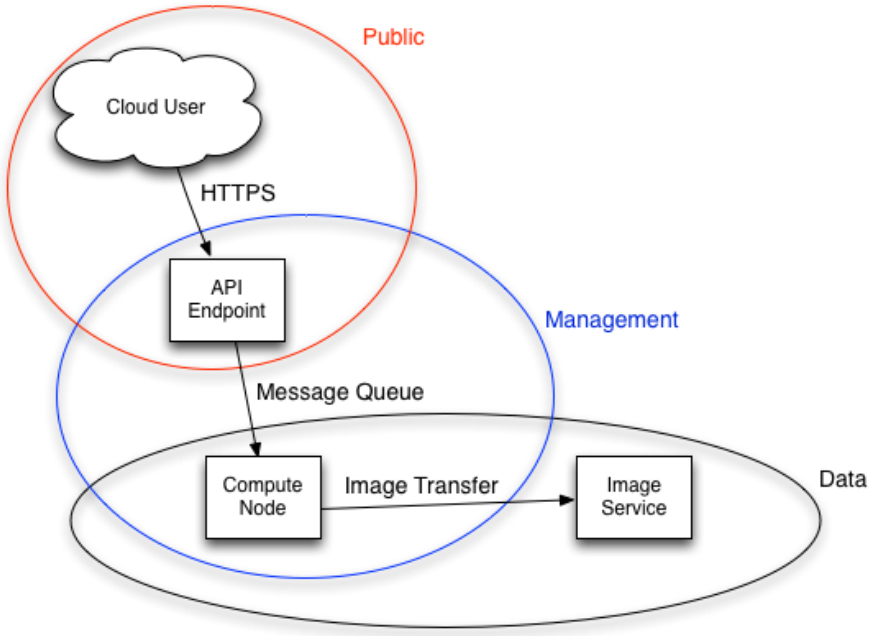
セキュリティドメインのブリッジ

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。

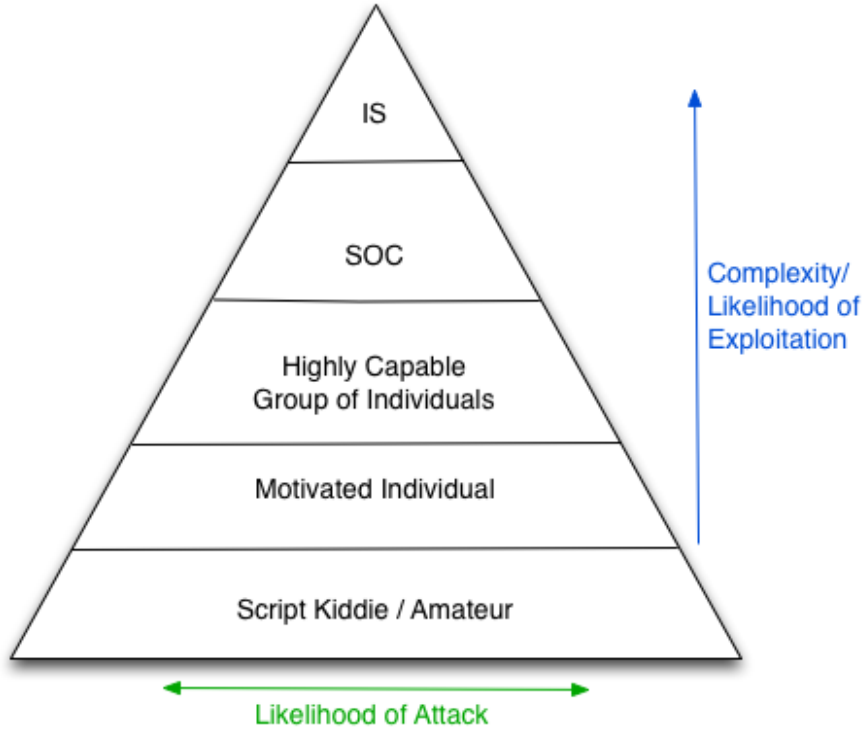
- インテリジェンスサービス — このガイドでは最も有能な攻撃者とされています。インテリジェンスサービスやその他の国家主体は、ターゲットに圧力をかけるために莫大なリソースを費やすことができます。他のどのアクターよりも能力があります。人、技術両方の面で非常に厳しい制御がないと、これらのアクターから防御することは極めて困難です。

- 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。

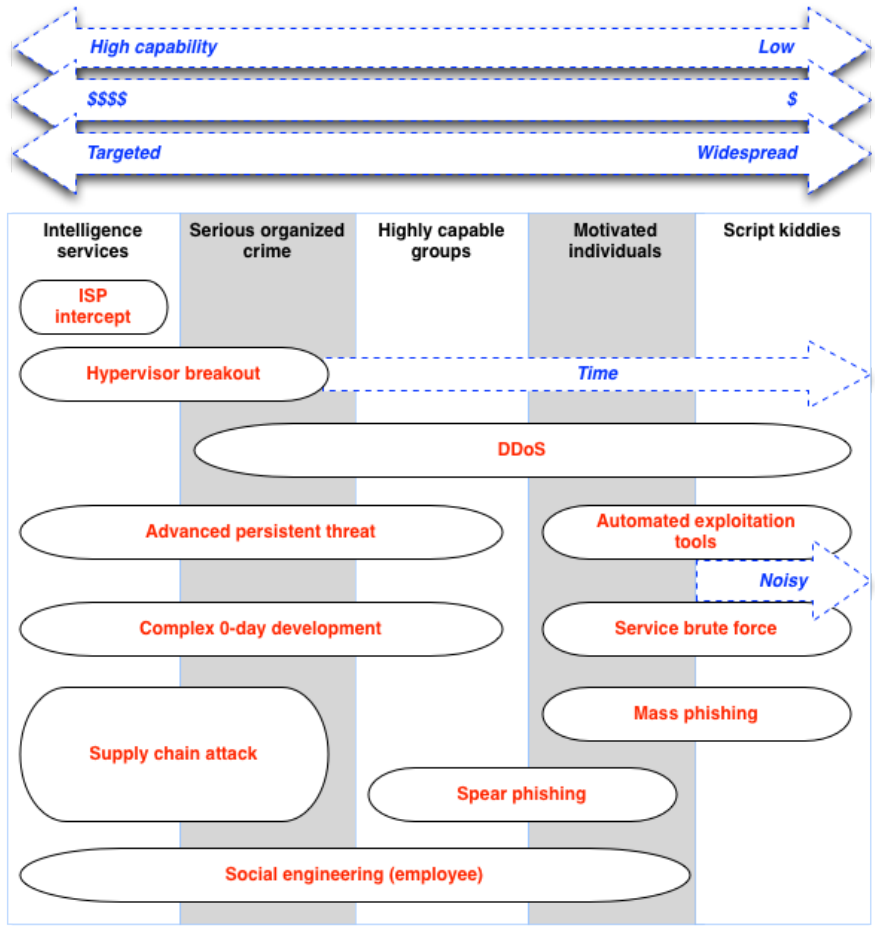
プライバシーの課題は、パブリッククラウドのユーザーとプライベートクラウドのユーザーとでは全く正反対になっています。プライベートクラウドで生成・格納されているデータは通常、データ損失防止（DLP）、ファイルの検査、ディープパケットインスペクション（DPI）、規範ファイアウォール（Prescriptive Firewall）などの技術をデプロイ可能なクラウドのオペレーターが所有します。反対に、パブリッククラウドには上記の様な制御の多くが存在しないため、プライバシーは、パブリッククラウドを採用する際の主な障害の 1 つとなっています。

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

本ガイドでは、全体を通して、2 つの運用事例を参照しています。本セクションでは、これらを概要を説明し、各章末で参照します。

事例: プライベートクラウドビルダーのアリス

アリスは、米国のある政府機関で使用するクラウドをデプロイしています。このクラウドは、FedRAMP などの関連基準に準拠する必要があり、またセキュリティ関連の文書業務を行う必要性が非常に高くなっています。クラウドは、インターネットには直接アクセスしてはなりません。API エンドポイント、Compute インスタンス、およびその他のリソースは、その政府機関のネットワーク内のシステムに対してのみ公開される必要があります。このネットワークは、他の全ネットワークから完全に隔離されています。クラウドは、この機関のイントラネット上で、認証/ロギングサービスなどの他のネットワークサービスにアクセスすることが可能です。

事例: パブリッククラウドプロバイダーのボブ

ボブは、新規展開の大規模なパブリッククラウドのデプロイを行う会社のリードアーキテクトです。このクラウドは、有効なクレジットカードを持つ消費者が、ユーティリティコンピューティングやストレージに使用できる一般大衆向けの IaaS を提供しますが、第一のターゲットは 企業顧客です。企業の間では、データプライバシー問題は、大規模なクラウド導入の大きな障害とみなされているため、ボブにとって優先課題となっています。

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

通常 OpenStack のインストールを構成している、広く定義された 2 つのノードタイプは次のとおりです。

- インフラストラクチャーノード。OpenStack Identity Service、メッセージキューサービス、ストレージ、ネットワーク、およびクラウドの運用をサポートするために必要なその他のサービスなどのクラウド関連サービスを実行するノードです。

- コンピュート、ストレージ、その他のリソースのノード。クラウド用のストレージ容量や仮想マシンを提供するノードです。

システムインベントリ

文書には、OpenStack 環境の概要を記載し、使用する全システム（実稼働、開発、テストなど）を対象とするべきです。多くの場合、システムコンポーネント、ネットワーク、サービス、およびソフトウェアについて文書化することにより、セキュリティ課題、攻撃ベクトル、考えられるセキュリティドメインのブリッジングポイントを完全に網羅して検討するにあたって必要な概観が提供されます。システムインベントリには、従来の IT システムでは永続的なリソースとされている、仮想マシンや仮想ディスクボリュームなどの一時的なリソースを取り込む必要がある場合があります。

ハードウェアインベントリ

文書化に対する厳密なコンプライアンス要件のないクラウドの場合は、少なくとも Configuration Management Database（CMDB）を使用することによってメリットが得られる可能性があります。CMDB は通常、ハードウェア資産の追跡や全般的なライフサイクル管理に使用されます。CMDB を活用することにより、組織はネットワーク上に存在するクラウドインフラストラクチャーハードウェア（例: コンピュートノード、ストレージノード、ネットワークデバイスなど）の中で適切に保護されていないハードウェアや忘れられているハードウェアを迅速に特定することができます。OpenStack のプロビジョニングシステムは、ハードウェア属性の自動検出機能が利用できる場合は特に、CMDB のような機能を一部提供することが可能です。

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

サービス、プロトコル、ポートの表には OpenStack デプロイメントの重要な追加情報を記載します。クラウドインフラストラクチャー内で稼働中の全サービスを表にまとめると、情報や指針を直ちに確認することが

でき、セキュリティプロシージャーをチェックするのに役立ちます。簡潔な情報が提供されると、ファイアウォールの設定やサービスポートの競合、セキュリティ修復領域、コンプライアンス要件をより容易に管理できるようになります。以下の表を検討してください。

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 ケーススタディ: システムのドキュメント

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、アップグレードが必要な時期を把握することも重要です。

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。これらだけではなく他のコンポーネントすべてを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となりますが、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が重要です。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

セキュリティ更新を通知された後、次のステップとして、指定のクラウドデプロイメントにとって、この更新がどの程度重要かを判断します。このような場合、ポリシーを事前定義しておくと便利です。共通脆弱性評価システム（CVSS）v2 などの既存の脆弱性評価システムは、クラウドデプロイメントに正しく対応していません。

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。

権限昇格とは、適切な認証チェックをすり抜けてシステム内の他のユーザーの権限を行使するユーザーの能力のことを指します。例えば、標準の Linux ユーザーがシステム上の root ユーザーの権限で自分の権限以上の操作を可能にするオペレーションを実行したり、コードを実行したりするなどです。

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

この表は、デプロイメントの発生箇所や影響をもとに脆弱性から受ける影響レベルを測定するための一般的な手法を示しています。例えば、Compute API ノードで権限レベルを 1 つ昇格すると、API の標準ユーザーはこのノード上の root ユーザーと同等の権限にまで昇格することが可能です。

クラウド管理者が、さまざまなセキュリティレベルに合わせて実行するアクションを定義する役に立てるために、この表をモデルとして使用することを推奨します。例えば、レベルが「重要」であるセキュリティ更新では、指定のスケジュールでクラウドのアップグレードが必要となる可能性がありますが、レベルが「低」の更新ではそこまで厳しくないでしょう。

更新のテスト

何かしらの更新を本番環境にデプロイする前に、それらをテストするようにしてください。一般的に、更新を最初に受信するテスト用のクラウド設定が別途必要になります。このクラウドのソフトウェアやハードウェアはできるだけ実稼働クラウドと同じ環境にする必要があります。パフォーマンスの影響、安定性、アプリケーションへの影響など、更新全体をテストする必要があります。特に重要なのは、更新で理論上対応されている問題（例: 特定の脆弱性）が実際に修正されているかどうかを確認することです。

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。

さらに、Git や SVN などのバージョン管理システムと統合すると、経年の環境の変化をトラッキングして、発生する可能性のある未認証の変更を修正することができます。例えば、nova.conf ファイルやその他の設定ファイルが規格に準拠しなくなった場合、既知の状態に構成管理ツールはファイルを復元または置き換えることができるでしょう。最後に、構成管理ツールを使用して、更新のデプロイも可能で、セキュリティパッチのプロセスを簡素化します。これらのツールには、この項において便利な機能が幅広く含まれています。クラウドのセキュリティ確保の主な目的は、構成管理のツールを選択して使用することです。

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
- セキュリティが強化された専用のバックアップサーバーを使用すること。バックアップサーバーのログは日次で監査し、ほんの一握りの人だけがこのログにアクセスできるようにしなければいけません。
- 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。

構成管理およびセキュリティ監査ツールは、もう１つのレベルで複雑性をクラウドにもたらします。この複雑性により、新たなセキュリティの課題が出てきます。これについては、セキュリティの利点もあるため、許容範囲のリスクのトレードオフという見解を持っています。これらのツールの運用におけるセキュリティ確保については、本書の対象外となっています。

第10章 完全性ライフサイクル

OpenStack では、完全性ライフサイクルを「クラウド全体にわたって想定されているソフトウェアが想定されている設定で常に実行されることを保証する計画的なプロセス」と定義しています。このプロセスは、セキュアなブートストラッピングで開始し、設定管理およびセキュリティ監視の機能により維持されます。本章では、完全性ライフサイクルプロセスのアプローチ方法について説明します。

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。

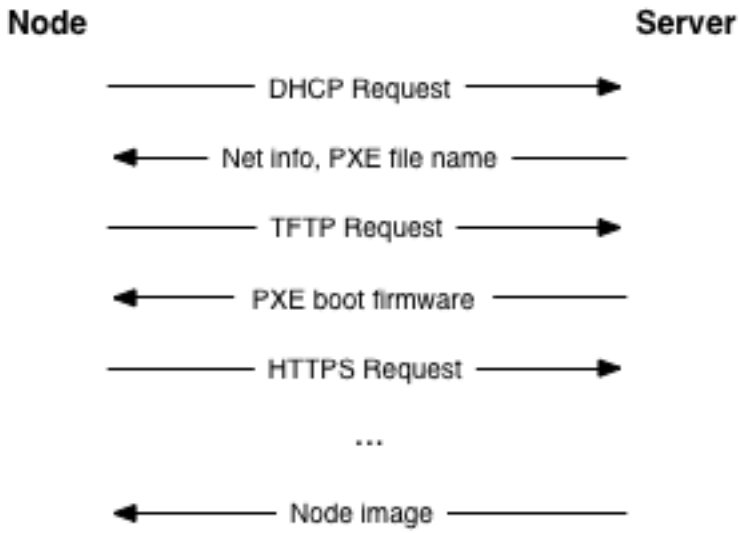
このような初期ブート段階の検証を可能にするさまざまな技術があります。通常は、Trusted Platform Module (TPM)、Intel Trusted Execution Technology (TXT)、Dynamic Root of Trust Measurement (DRTM)、Unified Extensible Firmware Interface (UEFI) などによるセキュアブートのハードウェアサポートが必要です。本ガイドでは、これらを総称してセキュアブートテクノロジーと呼びます。OpenStack ではセキュアブートの使用を推奨していますが、このデプロイに必要な諸作業には、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。

PXE ブートプロセス中に SSL を活用するのは若干困難ですが、iPXE などの一般的な PXE ファームウェアプロジェクトは、この機能をサポートしています。通常、この作業には、サーバーの証明書を適切に検証す

るための許可済み SSL 証明書チェーンについての知識を活用した PXE ファームウェア構築が伴います。これにより、安全性の低いプレーンテキストのネットワーク操作数が制限されるので、攻撃者に対するセキュリティレベルが高くなります。.

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) コード	ソフトウェア

	(例: マスターブートレコード)	
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

本ガイドの執筆時点では、実稼働環境でセキュアブートテクノロジーを使用するクラウドはほとんどありませんでした。このため、これらのテクノロジーはまだ若干未成熟な状態です。ハードウェアは、慎重に計画した上で選択することを推奨します (例: TPM および Intel TXT の対応を確認するなど)。次に、ノードのハードウェアベンダーが PCR 値をどのように事前設定しているかを検証します (例: どの値を検証できるか)。上記の表のコンテキストにソフトウェアと記載されている PCR 値は通常、クラウドアーキテクトが直接コントロールできます。ただし、これらの値は、クラウド内のソフトウェアをアップグレードすると変更される場合があります。設定管理は、PCR ポリシーエンジン内にリンクして、検証を常に最新の状態 に確保すべきです。

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。

上記のノードデプロイメントの戦略を前提とすると、Initial Program Loader (IPL) コードは、PXE ファームウェアである可能性が最も高く、このため、セキュアブートまたはブートアテステーションプロセスで、すべての初期段階のブートコード (例: BIOS、ファームウェアな

ど)、PXE ファームウェア、およびノードのカーネルを計測することができます。各ノードにこれらの正しいバージョンがインストールされていることを確認することにより、残りのノードソフトウェアスタックを構築する土台となる強固な基盤が提供されます。

選択した戦略に応じて、障害発生時にノードがブートに失敗するか、クラウド内の別のエンティティに障害を報告することができます。セキュアブートの場合には、ノードがブートに失敗し、管理セキュリティドメイン内のプロビジョニングサービスがこの問題を認識してイベントログを記録する必要があります。ブートアテステーションの場合には、障害検出時にはノードがすでに稼働している状態です。この場合、ネットワークアクセスを無効にすることによってノードの検疫を直ちに行った後に、イベントを解析して根本原因を特定するべきです。いずれの場合も、ポリシーにより、障害発生後の対処方法を指示する必要があります。クラウドが、特定の回数、ノードの再プロビジョニングを自動的に試みるようにしたり、問題を調査するようにクラウド管理者に直ちに通知するようにすることができます。この場合に適正となるポリシーは、デプロイメントと障害のモードによって異なります。

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。

- クラウドにネットワーク IDS の配置を検討することは重要です（例: ネットワーク境界や機密性の高いのネットワークに追加するなど）。配置はネットワーク環境によって異なりますが、追加する場所によって IDS がサービスにもたらす可能性のある影響を確実に監視するようにしてください。通常 ネットワーク IDS は、SSL などの暗号化トラフィックを調査することはできませんが、ネットワーク上の異常な非暗号化トラフィックを特定するメリットを提供することができます。
- 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 管理ユーティリティ (例: nova-manage, glance-manage)
- 帯域外管理インターフェース (IPMI など)

Dashboard

OpenStack dashboard (Horizon) は、管理者やプロジェクトに対して、クラウドベースのリソースのプロビジョンやアクセスができるようにWeb ベースのグラフィカルインターフェースを提供します。ダッシュボードは、上述の OpenStack API に呼び出しを行うことでバックエンドサービスと対話します。

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。

- また、Dashboard ではルーターやロードバランサーにも GUI 対応しています。例えば、Dashboard は主な Networking 機能をすべて実装するようになりました。
- 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

- Dashboard は Web ブラウザーのクッキーと JavaScript を有効にする必要があります。
- Dashboard をホストする Web サーバーは、データの暗号化が確実に行われるように SSL の設定をしてください。
- 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。
- （デプロイメント/セキュリティ関連の問題は多数ありますが）ダッシュボードでユーザーのハードディスクから OpenStack Image Service に直接イメージファイルをアップロードすることができるようになりました。サイズが GB レベルのイメージについては、Glace CLI を使用してイメージをアップロードするよう強く推奨しています。
- 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

- API はクラウド管理者がクラウドデプロイメントのサイズや状態の概要を把握できるようにするだけでなく、ユーザー、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、プロジェクト（テナント）ベースのリソースクォータの指定などができるようにします。
- 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。

管理ユーティリティ

OpenStack 管理ユーテリティは、API 呼び出しを行う、オープンソースの Python のコマンドラインクライアントです。OpenStack サービス（nova、glance など）毎にクライアントがあります。標準の CLI クライアントに加え、サービスの多くには管理コマンドラインがあり、データベースへ直接呼び出しを行います。これらの専用の管理ユーテリティは徐々に廃止予定となっています。

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 エンドユーザーガイド の項: コマンドラインクライアントの概要

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 コンポーネントを実行するノードにアクセスする場合、OpenStack の管理は IPMI プロトコルなどの帯域外管理インター

フェースに依存します。IPMI は非常に有名な仕様で、オペレーティングシステムの実行中である場合やシステムがクラッシュした場合でもリモートでのサーバー管理、診断、リブートを行います。

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。
- Web インターフェースを使用して BMC/IPMI と対話する場合、常に SSL インターフェースを使用するようにしてください（例: https またはポート 443)。この SSL インターフェースは自己署名証明書を使用しないようにしてください。通常、これがデフォルトとなっていますが、正しく定義された完全修飾ドメイン名（FQDN）を使用して信頼済みの証明書を使用するようにしてください。
- 管理ネットワークのトラフィックを監視します。トラフィックの多い Compute ノードよりも例外のトラッキングが簡単になる場合があります。

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 ケーススタディ: 管理インターフェース

一般的な OpenStack 管理インターフェースと関連のバックプレーンの問題について、ここまでに議論しました。再度、アリスとボブのケーススタディに戻って、これらの問題を見ていきます。特に、アリスとボブが以下の点をどのように対応したかを確認していきます。

- 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
- Dashboard

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 ケーススタディ: PKI と証明書管理

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

以下に、幾つかの主な有名 Web サーバ／SSL 終端を推奨設定でSSLを有効にする為の幾つかの設定例を示します。クライアント互換性の為に多くのデプロイで必要になる筈なので、幾つかの例ではSSL v3 が有効になっている点に注意して下さい。

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-accelleration 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: This list does not include any RC4 ciphers.
ciphers = ”HIGH:!aNULL:!eNULL:!DES:!3DES”
# 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:!aNULL:!eNULL:!DES:!3DES;

    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:!aNULL:!eNULL:!DES:!3DES;
  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:!aNULL:!eNULL:!DES:!3DES;
```

```
  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

Identity Service のカタログは内部 URL を認識できるようにすべきです。この機能はデフォルトで利用されませんが、設定により有効化できます。さらに、この動作が標準になると、予期される変更と前方互換性があるべきです。

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 ケーススタディ: API エンドポイント

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

OpenStack Identity Service (Keystone) は、ユーザー名・パスワード、LDAP、外部認証方式など、複数の認証方式をサポートします。認証に成功すると、Identity Service は後続のサービスリクエストに使用する認可トークンをユーザーに返します。

Transport Layer Security TLS/SSL は、サービスと人の間で X.509 を使用した認証を提供します。SSL の規定のモードはサーバーのみを認証しますが、証明書はクライアントを認証するためにも使用されるかもしれません。

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。

潜在的な DoS 攻撃に関する OpenStack Security Note (OSSN) があります。

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。
- 多要素認証: 認証サービスが、ユーザーが持っているもの（例: ワンタイムパスワードトークン、X.509 証明書）と知っていること（例: パスワード）に基づいた情報を提示するよう要求します。
- 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。

Identity Service は SSL のクライアント認証を有効化していると、それをサポートします。SSL クライアント認証はユーザー名、パスワードに加えて、ユーザー識別により信頼性を与えるために追加の認証要素を提供します。ユーザー名とパスワードが漏えいした場合に、権限のないアクセスのリスクを減らすことができます。しかしながら、証明書を

ユーザーに発行する追加の管理作業とコストが発生します。これはすべての環境で実現できるとは限りません。

注記

Identity Service にサービスの認証をするとき、SSL を使用したクライアント認証を使用することを推奨します。

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。

SSL を用いたクライアント認証のために、証明書を発行する必要があります。これらの証明書は外部の認証局やクラウド管理者により署名できます。OpenStack のサービスはデフォルトで証明書の署名を確認します。証明が確認できなければ、接続に失敗します。管理者が自己署名証明書を使用している場合、確認を無効化する必要があるかもしれません。これらの証明書を無効化するために、/etc/nova/api.paste.ini の [filter:authtoken] セクションに insecure=False を設定します。これらの設定は他のコンポーネントの証明書も無効化します。

管理ユーザー

管理ユーザーは Identity Service や証明書のような 2 要素認証をサポートする外部認証サービスを使用して認証することを推奨します。これにより、パスワード推測によるリスクを減らすことができます。この推奨事項は特権アカウントへのネットワークアクセスに多要素認証を使用するという NIST 800-53 IA-2(1) ガイドに適合しています。

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。

以下は PKI トークンの例です。実際は token id の値が約3500バイトであることに注意してください。この例では短くしています。

```
{
   "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"
      }
   }
}
```

トークンは Identity Service 応答のより大きなコンテキスト構造の中で渡されることに注意してください。これらの応答はさまざまな

OpenStack サービスのカタログも提供しています。各サービスはその名前と、内部、管理、パブリックなアクセス用のエンドポイントを一覧にします。

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 Dashboard

Horizon は OpenStack のダッシュボードです。管理者により設定された制限の範囲内でユーザー自身のリソースを展開できるセルフサービスポータルをユーザーに提供します。これらには、ユーザーの管理、インスタンスのフレーバーの定義、仮想マシンイメージのアップロード、ネットワークの管理、セキュリティグループのセットアップ、インスタンスの起動、インスタンスへのコンソール経由のアクセスなどがあります。

ダッシュボードは Django ウェブフレームワークに基づいています。そのため、Django のセキュアな導入プラクティスをそのまま Horizon に適用できます。このガイドは Django のセキュリティ推奨事項の一般的なものを提供します。さらなる情報は Django deployment and security documentation を読むことにより得られます。

ダッシュボードは適度なデフォルトのセキュリティ設定をしてあります。また、素晴らしい deployment and configuration documentation (導入と設定のドキュメント) があります。

基本的なウェブサーバーの設定

ダッシュボードは、Apache や nginx のような HTTPS プロキシの後ろに Web Services Gateway Interface (WSGI) アプリケーションとして導

入すべきです。まだ Apache を使用していなければ、nginx を推奨します。こちらのほうが軽量かつ正しく設定しやすいです。

nginx を使用している場合、適切な数の同期ワーカーを持つ WSGI ホストとして gunicorn を推奨します。fastcgi、scgi または uWSGI を使用して導入することを強く推奨します。WSGI サーバーを選択するとき、統合パフォーマンスベンチマークを使用することを強く推奨します。

Apache を使用しているとき、ダッシュボードをホストするために mod_wsgi を推奨します。

HTTPS

ダッシュボードは、認知されている認証局 (CA) から発行された有効かつ信頼できる証明書を使用しているセキュアな HTTPS サーバーの後ろに導入すべきです。プライベートな組織で発行された証明書は、ルート証明機関がお使いのすべてのブラウザーに事前インストールされているときのみ、適切に動作します。

ダッシュボードのドメインに対する HTTP リクエストは、完全修飾された HTTPS URL にリダイレクトされるよう設定すべきです。

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。

注: ウェブブラウザーの前で HTTPS プロキシを使用している場合、HTTPS 機能を持つ HTTP サーバーを使用するより、Django documentation on modifying the SECURE_PROXY_SSL_HEADER variable に従うほうが良いです。

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。

フロントエンドキャッシュ

ダッシュボードは OpenStack API リクエストから渡された動的コンテンツをそのまま描画するため、varnish のようなフロントエンドキャッシュ層を推奨しません。Django では、静的なメディアは直接 Apache や nginx から処理され、すでにウェブホストのキャッシュの恩恵を受けています。

ドメイン名

多くの組織は一般的に、組織全体のドメインのサブドメインにウェブアプリケーションを配備します。ユーザーが openstack.example.org 形式のドメインを期待することは自然です。これに関連して、しばしば同じ第 2 レベルの名前空間に配備された、ユーザーが管理できるコンテンツを取り扱う他の多くのアプリケーションがあります。この名前の構造は便利であり、ネームサーバーのメンテナンスを簡単にします。

Horizon を第 2 レベルドメインに導入することを強く推奨します。たとえば、https://example.com です。また、Horizon を共有サブドメインに導入しないことをお奨めします。たとえば、https://openstack.example.org や https://horizon.openstack.example.org です。https://horizon/ のようなそのままの内部ドメインに導入しないこともお奨めします。

この推奨事項はブラウザーの同一オリジンポリシーの制限に基づいています。このガイドにある推奨事項は、次のような場合によく知られている攻撃からユーザーを効率的に保護できません。ユーザーが生成したコンテンツ（例: スクリプト、イメージ、あらゆる種類のアップロード）もホストしているドメインにダッシュボードを導入した場合です。ユーザーが生成したコンテンツが別のサブドメインにある場合もです。この方法は、ユーザーが生成したコンテンツをクッキーやセキュリティトークンから確実に分離するために、多くの有名なウェブサイト（例: googleusercontent.com、fbcdn.com、github.io、twimg.com）により使用されています。

さらに、第 2 レベルドメインに関する上の推奨事項に従わない場合、クッキーによるバックエンドセッションを避け、HTTP Strict Transport Security (HSTS) を採用することがきわめて重要です。サブドメインに導入するとき、ダッシュボードのセキュリティは同じレベルのドメインに導入されているアプリケーションの中で最も弱いレベルと同じ強度になります。

静的メディア

ダッシュボードの静的メディアは、ダッシュボードのドメインのサブドメインに導入し、ウェブサーバーにより処理されるべきです。外部の CDN (content delivery network) の使用も問題ありません。このサブドメインは、クッキーを設定すべきではなく、ユーザーが提供したコンテンツを処理すべきではありません。メディアは HTTPS を用いても処理されるでしょう。

Django のメディア設定は https://docs.djangoproject.com/en/1.5/ref/settings/#static-root にドキュメント化されています。

ダッシュボードのデフォルトの設定は、CSS と JavaScript のコンテンツを処理する前に圧縮して最小化するために django_compressor を使用します。このプロセスは、デフォルトのリクエストごとに動的な圧縮を使用する代わりに、ダッシュボードを導入して、導入されたコードと一緒に結果のファイルを CDN サーバーにコピーする前に静的に実行されるべきです。圧縮は本番環境以外で実行すべきです。これが実践的でなければ、すべてのリソースの圧縮を無効化することを推奨します。オンライン圧縮の依存物（less や nodejs）は本番環境にインストールするべきではありません。

シークレットキー

ダッシュボードはいくつかのセキュリティ機能に関する共有 SECRET_KEY 設定に依存します。これはランダムに生成された最小 64 文字の文字列です。すべての Horizon インスタンスで共有する必要があります。このキーが漏洩すると、リモートの攻撃者が任意のコードを実行できる可能性があります。このキーのローテーションにより、既存のユーザーセッションとキャッシュを無効化します。このキーを公開リポジトリにコミットしないでください。

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。

お使いのアーキテクチャーが許容できる場合、セッションバックエンドとして django.contrib.sessions.backends.cache を、キャッシュとして memcache を一緒に使用することを推奨します。memcache はパブリックにアクセスされてはいけません。セキュアなプライベートチャネル経由で通信すべきです。署名付きクッキーバックエンドを使用することにした場合、セキュリティのトレードオフを理解するために Django のドキュメントを参照してください。

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

Horizon が利用可能なドメインを ALLOWED_HOSTS に設定します。この設定を失敗すると（とくに第 2 レベルドメインに関する上の推奨に従わなかった場合)、Horizon がさまざまな深刻な攻撃にさらされます。ワイルドカードを使用したドメインは避けるべきです。

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
```

ドットから始まるワイルドカードドメインを持つよう、CSRF やセッションクッキーを設定してはいけません。Horizon のセッションクッキーと CSRF クッキーは HTTPS を使用した環境のときにセキュア化すべきです。

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

パスワード自動補完

実装者は標準のパスワードオートコンプリート機能を変更しないことを推奨します。ユーザーはセキュアなブラウザのパスワードマネージャーを使用できる環境で、より強力なパスワードを選択します。ブラウザのパスワードマネージャーを禁止している組織は、デスクトップレベルでこのポリシーを強制すべきです。

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。

ダッシュボードは、カスタマイズしたダッシュボードでクロスサイトスクリプティングの脆弱性が含まれることから、開発者を守るよう設計されています。しかしながら、カスタマイズしたダッシュボード、とくに、@csrf_exempt デコレーターを不適切に使用して javascript を多用しているものを監査することは重要です。これらのセキュリティ設定の推奨事項に従わないダッシュボードは、制限を緩和する前に注意深く評価されるべきです。

クロスサイトスクリプティング (XSS)

多くの似たようなシステムと異なり、OpenStack のダッシュボードは多くの項目にすべての Unicode 文字を許可します。このことは、開発者が XSS 攻撃の余地を残すエスケープミスをする範囲が少なくなることを意味します。

ダッシュボードは開発者が XSS 脆弱性を作ることを防ぐためのツールを提供します。しかし、それらは開発者が適切に使用するときのみ機能します。mark_safe 関数の使用、カスタムテンプレートタグを持つ is_safe の使用、safe テンプレートタグ、自動エスケープが無効化されているすべての場所、不適切にエスケープされたデータを評価するすべての javaScript にとくに注意して、すべてのカスタムダッシュボードを監視します。

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/
```

ワイルドカードオリジンを許可してはいけません。

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 セキュリティリリースは、一般的に十分にテストされ、積極的に後方互換性を確保しています。ほぼすべての場合、Django の新しいメジャーリリースも前のリリースと後方互換性があります。ダッシュボードの実装者は、最新のセキュリティリリースを持つ最新の安定リリースの Django を実行することを強く推奨されます。

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

クラウドアーキテクトが判断する必要があることの一つは、Compute Service の設定が VNC と SPICE のどちらを使用するかです。以下は、これらの選択肢の違いに関する詳細を提供します。

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。
- nova コマンドラインユーティリティは SPICE-html クライアントによりアクセスするための SPICE コンソールの URL を返すことができます。

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) は HTTP 経由でデータの保存と取得を提供するサービスです。オブジェクト (データの小さな塊) は、認証機構に基づいて匿名の読み込み専用アクセス権や ACL 定義のアクセス権を提供する組織化した階層に保存されます。

利用者は、オブジェクトの保存、それらの変更、HTTP プロトコルと REST API を使用したアクセスを実行できます。Object Storage のバックエンドコンポーネントは、サービスの冗長化クラスターで同期された情報を維持するために別のプロトコルを使用します。API とバックエンドコンポーネントの詳細は OpenStack Storage のドキュメントを参照してください。

このドキュメントの場合、コンポーネントは以下の主要なグループに分けています。

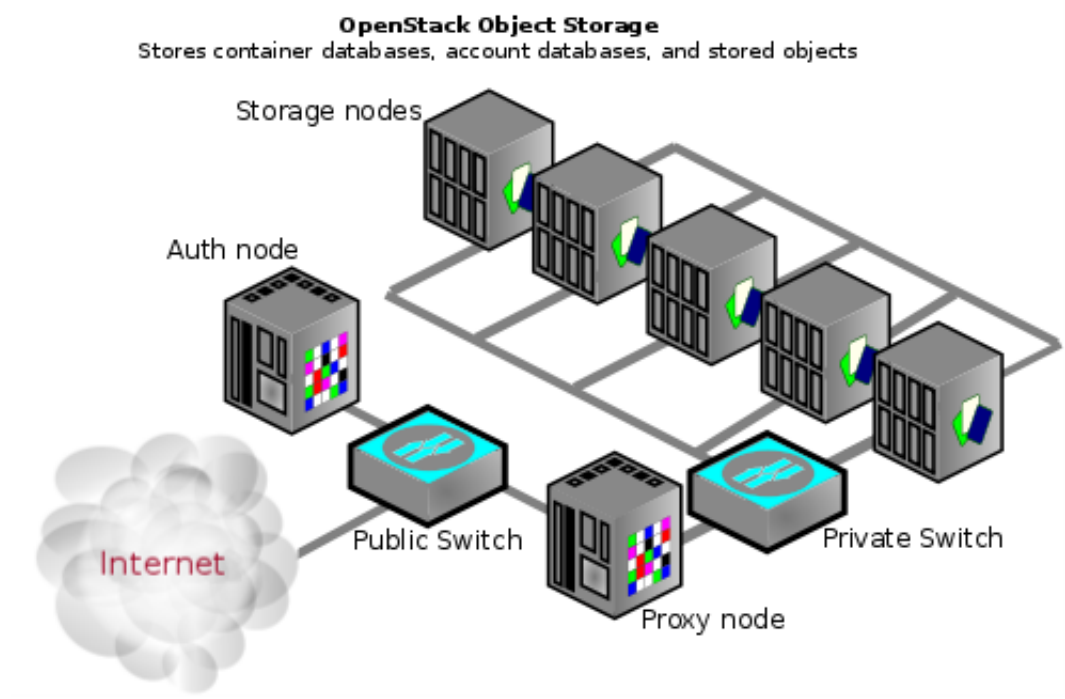
1. プロキシサービス

2. 認証サービス

3. ストレージサービス

   - 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 OpenStack Object Storage Administration Guide (2013) からのサンプル図



注記

Object Storage 環境はインターネット環境にある必要がありません。組織の内部ネットワークインフラストラクチャーの一部である「パブリックスイッチ」を使用してプライベートクラウドにできます。

最初にセキュア化するもの - ネットワーク

Object Storage に対するセキュアなアーキテクチャー設計の最初の観点はネットワークコンポーネントです。ストレージサービスノードは、データの複製と高可用性を提供するためにお互いにデータをコピーするために rsync を使用します。プロキシサービスはさらに、データをバックエンドと中継するとき、そして 4 つ目にエンドポイントのクライアントとクラウド環境の間で中継するときに、ストレージサービスと通信します。

注意

これらはこの階層で何も暗号化や認証を使用しません。

これがアーキテクチャー図に「プライベートスイッチ」やプライベートネットワーク（[V]LAN）が書かれている理由です。このデータドメインは他の OpenStack データネットワークと分離すべきです。セキュリティドメインにおけるさらなる議論は 4章セキュリティ境界と脅威［15］を参照してください。

ヒント

ルール: データドメインでストレージサービスのためにプライベート（V)LAN ネットワークを使用します。

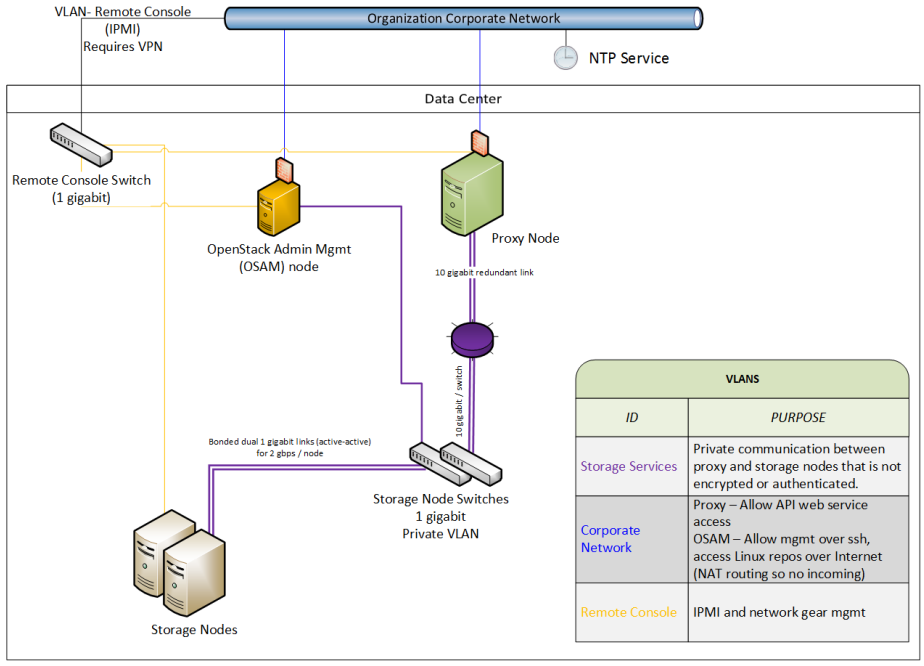
これにより、プロキシサービスノードが 2 つのインターフェース（物理または仮想）を持つ必要があります。

1. 利用者が到達できる「パブリック」インターフェースとして一つ

2. ストレージノードにアクセスする「プライベート」インターフェースとしてもう一つ

以下の図はある実現可能なネットワークアーキテクチャーを説明します。

図21.2 マネジメントノードを持つオブジェクトストレージネットワークアーキテクチャー (OSAM: Object storage network architecture with a management node)



サービスのセキュア化 - 一般

ユーザーとして実行するサービス

各サービスを root (UID 0) 以外のサービスアカウントで実行するよう設定することを推奨します。ある推奨事項はユーザー名「swift」と主グループ「swift」とすることです。

ファイルパーミッション

/etc/swift はリングのトポロジーと環境設定に関する情報を含みます。以下のパーミッションが推奨されます。

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

これは、サービスが「swift」グループメンバーに読み込むことを許可しながら、root のみが設定ファイルを変更できるように制限します。

ストレージサービスのセキュア化

以下はさまざまなストレージサービスのデフォルトのリッスンポートです。

サービス名	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

認証はこのレベルで Object Storage にありません。誰かがアクセスできるこれらのポートのどれかでストレージサービスノードに接続できる場合、認証なしでデータを変更できます。この問題に対してセキュアにするために、プライベートストレージネットワークを使用することに関して前に説明した推奨事項に従うべきです。

オブジェクトストレージの「アカウント」という用語

オブジェクトストレージの「アカウント」はユーザーアカウントやクレデンシャルではありません。以下に関連を説明します。

OpenStack Object Storage アカウント	コンテナーの集合体。ユーザーアカウントや認証ではありません。どのユーザーがアカウントに関連づけられるか、どのようにアクセスできるかは、使用する認証システムに依存します。後から認証システムを参照してください。このドキュメントで OSSAccount として参照されます。
OpenStack Object Storage コンテナー	オブジェクトの集合体。コンテナーにあるメタデータは ACL が利用可能です。ACL の意味は使用する認証システムに依存します。
OpenStack Object Storage オブジェクト	実際のデータオブジェクト。オブジェクトレベルの ACL はメタデータ付きでも可能です。これは使用する認証システムに依存します。

ヒント

上のことについて考える別の方法です。一つの書庫（アカウント）0 またはそれ以上の入れ物（コンテナー）を持ちます。入れ物（コンテナー）はそれぞれ 0 またはそれ以上のオブジェクトを持ちます。車庫（Object Storage クラウド環

境）は、それぞれ 0 またはそれ以上のユーザーが所属する書庫（アカウント）を複数持つ可能性があります。

各レベルに、誰がどの種類のアクセス権を持つのかを記録する ACL を持つかもしれません。ACL はどの認証システムが使用されているのかに依存して解釈されます。最も一般的に使用される 2 種類の認証プロバイダーは Keystone と SWAuth です。カスタム認証プロバイダーも利用できます。詳細は Object Storage 認証のセクションを参照してください。

プロキシサービスのセキュア化

プロキシサービスノードは少なくとも 2 つのインターフェース（物理または仮想）を持つべきです。一つはパブリック、もう一つはプライベートです。パブリックインターフェースはファイアウォールやサービスバインディング経由で保護できるかもしれません。パブリックなサービスは、エンドポイントクライアントのリクエストを処理し、それらを認証し、適切なアクションを実行する HTTP ウェブサーバーです。プライベートインターフェースはサービスをリッスンしませんが、代わりにプライベートストレージネットワークにあるストレージサービスノードに接続を確立するために使用されます。

SSL/TLS の使用

Swift に組み込みまたは同梱されているウェブサーバーは SSL をサポートします。しかし、SSL 証明書チェイン全体の送信をサポートしません。これにより、お使いのクラウド用に Verisign のような第三者機関により信頼されて署名された証明書を使用するときに問題を引き起こします。現在の回避策は組み込みのウェブサーバーを使用せず、公開サーバー証明書と CA 中間認証局の証明書の両方を送信することをサポートする別のウェブサーバーを代わりに使用することです。これにより、エンドポイントのクライアントがお使いのクラウド環境の SSL 証明書とチェインを正常に検証できるようになるために、それらの信頼ストアにある CA ルート証明書を持てるようになります。mod_wsgi と Apache を用いてこのようにする方法の例が以下にあります。また、Apache Deployment Guide を参照してください。

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

次のように /etc/apache2/envvars ファイルを変更します。

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

別の方法は Apache の設定ファイルを次のように変更することです。

```
            User swift
            Group swift
```

Apache のドキュメントルートに「swift」ディレクトリを作成します。

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

$YOUR_APACHE_DOC_ROOT/swift/proxy-server.wsgi ファイルを作成します。

```
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

これまでに説明したように「swift」のように非 root ユーザー（UID 0 以外）としてプロキシサービスのウェブサーバーを実行すべきです。これを簡単にし、何らかのウェブコンテナーの部分を root として実行することを避けるために、1024 より大きいポートを使用することが必要です。エンドポイントのクライアントは一般的にオブジェクトストレージをブラウジングするためにウェブブラウザーに手動で URL を入力することがないため、そのようにすることは大変でありません。さらに、HTTP REST API を使用して、認証を実行するクライアントに対して、認証のレスポンスにより提供されるとおり、使用する完全な REST API URL を通常は自動的に取ってきます。OpenStack の REST API により、クライアントがある URL に認証できるようになり、実際のサービスのために別の URL を使用するようにできます。例: クライアントが https://identity.cloud.example.org:55443/v1/auth に認証して、それらの認証キーを持つ応答とストレージの URL（プロキシノードまたは負荷分散装置の URL）https://swift.cloud.example.org:44443/v1/AUTH_8980 を取得します。

ウェブサーバーを root 以外のユーザーで起動して実行する設定方法はウェブサーバーと OS により異なります。

負荷分散装置

Apache を使用するという選択肢が実現できない場合、またはパフォーマンスのために SSL 処理をオフロードしたい場合、専用のネットワークデバイスの負荷分散装置を使用できます。これは、複数のプロキシノード

を使用するときに、冗長性と負荷分散を提供するために一般的な方法です。

SSL をオフロードすることにした場合、ネットワーク上の他のノード（侵入されているかもしれない）が暗号化されていない通信を盗聴できないように、負荷分散装置とプロキシノード間のネットワークリンクは必ずプライベート (V)LAN セグメントに置くべきです。そのようなセキュリティ侵害が発生した場合、攻撃者はエンドポイントクライアントやクラウド管理者のクレデンシャルのアクセス権を取得し、クラウドのデータにアクセスできます。

Keystone や SWAuth のような使用する認証サービスが、エンドのクライアントへの応答にあるそれぞれの URL をどのように設定するのかを判断します。そのため、それぞれのプロキシサービスノードの代わりに、お使いの負荷分散装置を使用します。

オブジェクトストレージ認証

Object Storage はエンドポイントクライアントを認証するためのミドルウェアを提供するために wsgi を使用します。認証プロバイダーはどのロールとユーザー種別が存在するかを定義します。いくつかは伝統的なユーザー名とパスワードのクレデンシャルを使用します。一方、他のものは API キートークンやクライアントサイド x.509 SSL 証明書を活用します。カスタムプロバイダーは wsgi モデルを使用して統合できます。

Keystone

Keystone が OpenStack で一般的に使用される認証プロバイダーです。これは Object Storage でも認証のために使用できます。Keystone のセキュア化についてはすでに 18章Identity [75] で提供されています。

SWAuth

SWAuth は Keystone の代替となるものです。Keystone と比較して、オブジェクトストレージ自体にユーザーアカウント、クレデンシャル、メタデータを保存します。詳細は SWAuth のウェブサイト http://gholt.github.io/swauth/ にあります。

他の重要事項

すべてのサービスノードの /etc/swift/swift.conf に「swift_hash_path_suffix」設定があります。保存されているオブジェ

クトに対するハッシュ衝突の可能性を減らし、あるユーザーが別のユーザーのデータを上書きすることを防ぐために、これが提供されます。

この値は、暗号学的に安全な乱数生成器を用いて初期設定され、すべてのサービスノードにわたり一貫性を持つべきです。適切な ACL を用いて確実に保護され、データ損失を避けるためにバックアップコピーを必ず持つべきです。

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。

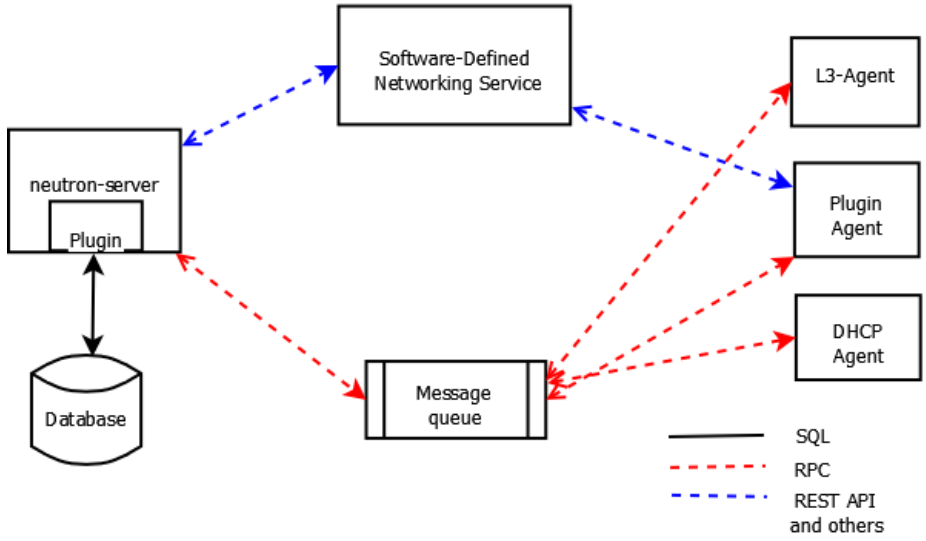
- プラグインエージェント (neutron-*-agent): ローカルの仮想スイッチ (vswitch) 設定を管理する為に各 compute ノード上で実行されます。実行するエージェントは、あなたが使用するプラグインに依存するでしょう。このサービスはメッセージキューへのアクセスを必要とします。オプションのプラグインに依存します。

- 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。

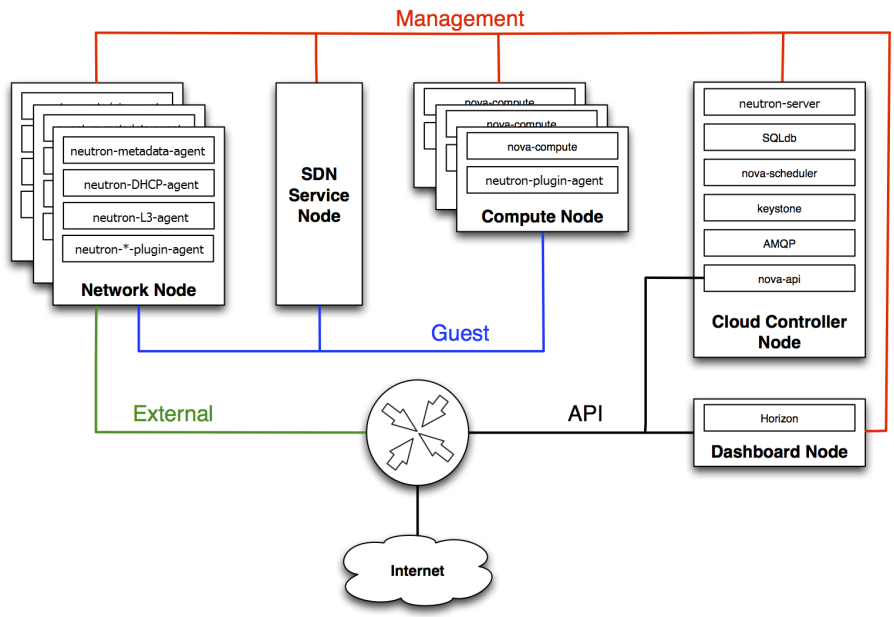
次表はOpenStack Networking コンポーネント群の構造・ネットワークフローダイアグラムを示しています。



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 プラグイン、Brocade Neutron プラグイン、Cisco UCS/Nexus プラグイン、Cloudbase Hyper-V プラグイン、Extreme Networks プラグイン、Juniper Networks Neutron プ

ラグイン、Linux Bridge プラグイン、Mellanox Neutron プラグイン、MidoNet プラグイン、NEC OpenFlow プラグイン、Open vSwitch プラグイン、PLUMgrid プラグイン、Ruijie Networks プラグイン、Ryu OpenFlow Controller プラグイン、VMware NSX プラグイン

Folsom リリース時点でプラグイン群が提供する全ての機能の詳細な比較表は、Sebastien Han's comparison を参照して下さい。

Networking サービスの制限事項

OpenStack Networking は以下の制限があります。

- IP アドレス重複 — neutron-l3-agent か neutron-dhcp-agent のいずれかを実行するノードが重複した IP アドレスを使用する場合、これらのノード群は Linux のネットワークネームスペースを使用する必要があります。デフォルトでは、DHCP と L3 エージェントは Linux ネットワークネームスペースを使用しています。しかしながら、ホストがこのネームスペースをサポートしていない場合、DHCP と L3 エージェントは異なるホストで実行して下さい。

  ネットワークネームスペースサポートがない場合、L3エージェントでは追加の制限事項として単一の論理ルータのみサポートされます。

- 複数ホスト DHCP エージェント — OpenStack Networking は複数の L3 エージェントと DHCP エージェントによる負荷分散をサポートしています。しかしながら、（訳注：nova-network がサポートしていた）仮想マシンとの配置上の強い紐付けはサポートされていません。

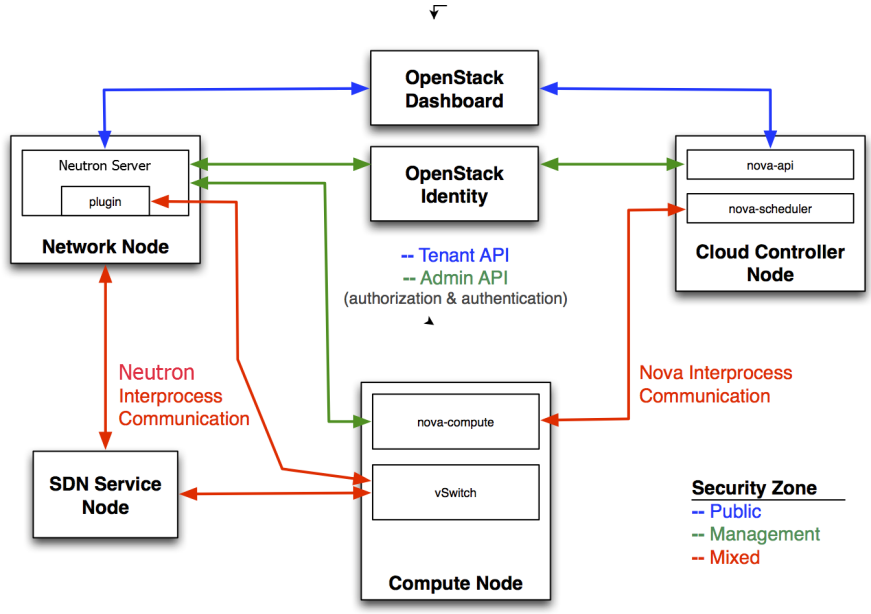
- L3 エージェントの IPv6 未対応 — neutron-l3-agent （L3 転送の実装用に多くのプラグインが使用）は IPv4 転送のみサポートしています。

第26章 OpenStack Networking サービスのセキュリティ強化

OpenStack Networking のセキュリティを強化する為に、テナントインスタンス作成用のワークフロープロセスの理解をセキュリティドメインにマッピングさせる必要があります。

OpenStack Networking と交信する主要なサービスが４つあります。典型的な OpenStack デプロイでは、これらのサービスは以下のセキュリティドメインにマッピングされます。

- OpenStack Dashboard: パブリック、管理
- OpenStack Identity: 管理
- OpenStack Compute ノード: 管理、ゲスト
- OpenStack ネットワークノード: 管理、ゲスト（使用する neutron プラグインによってはパブリックも可能性あり）
- SDB サービスノード：管理、ゲスト （使用する製品によってはパブリックも可能性あり）



OpenStack Networking サービスと他の OpenStack コアサービス間の扱いの難しいデータ通信を分離する為、通信を独立した管理ネットワーク上でのみ行うように通信路を設定する事を強く推奨します。

OpenStack Networking サービス設定

API サーバがバインドするアドレスの制限: neutron-server

OpenStack Networking API サービスが外からのクライアント通信用にネットワークソケットをバインドするネットワークインターフェース又は IP アドレスを制限する為、neutron.conf ファイル中の bind_host と bind_port を以下のように指定します。

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

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

OpenStack Networking サービス群の DB と RPC 通信の制限:

OpenStack Networking サービスの様々なコンポーネントは、OpenStack Networking 中の他のコンポーネントとの通信にメッセージキュー又はデータベース接続のいずれかを使用します。

DB への直接接続を必要とする全てのコンポーネントに対し、データベースの章のデータベース認証とアクセスコントロールの節で示されたガイドラインに従う事を推奨します。

RPC 通信を必要とする全てのコンポーネントに対し、メッセージングの章のキュー認証とアクセスコントロールの節中で示されたガイドラインに従う事を推奨します。

第27章 Networkingサービス セキュリティベストプラクティス

この章では、あなたの OpenStack デプロイの中でテナントネットワークセキュリティを適用する為に、OpenStack Networking の設定のベストプラクティスについて議論します。

テナントネットワークサービスのワークフロー

OpenStack Networking は、ネットワークリソースと設定の本物のセルフサービスをユーザに提供します。クラウドアーキテクトとオペレータが、利用可能なネットワークリソースの作成・更新・削除機能をユーザに提供する際の彼らの設計ユースケースを評価する事は重要です。

Networking リソースポリシーエンジン

OpenStack Networking 中のポリシーエンジンとその設定ファイル（policy.json）は、テナントネットワークメソッドとオブジェクト上のユーザのきめ細かな許可を提供する方法を提供します。クラウドアーキテクトとオペレータが、ユーザとテナントに利用可能なネットワークリソースを作成・交信・削除する機能を提供するにあたって、かれらの設計とユースケースを評価する事は重要です。なぜなら、テナントネットワークの可用性、ネットワークセキュリティ、全般的な OpenStack セキュリティ上でこれらが実際の効果を持つからです。OpenStack Networking ポリシー定義のより詳細な説明は、OpenStack クラウド管理者ガイド 中の 認証と認可の章を参照して下さい。

デフォルトの Networking リソースポリシーをレビューする事と、あなたのセキュリ

あなたの OpenStack のデプロイが異なるセキュリティドメインに向けて複数の外部アクセスポイントを提供する場合、複数の外部アクセスポイントへ複数の仮想NICをアタッチするテナントの機能を制限する事は重要

です。これは、これらのセキュリティドメインのブリッジになり、思いがけないセキュリティの妥協を導くかも知れません。OpenStack Compute が提供するホストアグリゲート機能の活用や、異なる仮想ネットワーク設定を持つ複数のテナントプロジェクトにテナントVMを分割する事で、リスクを緩和する事が可能です。

セキュリティグループ

OpenStack Networking サービスは、OpenStack Compute 上に構築されたセキュリティグループ機能より柔軟で協力な機能を用いたセキュリティグループ機能を提供します。このように、OpenStack Networking を用いる場合、nova.conf は常にビルトインのセキュリティグループを無効化し、全てのセキュリティグループ要求を OpenStack Networking API にプロキシする必要があります。これを怠った場合、セキュリティポリシーが両サービスに同時に適用されて衝突を起こす結果となります。OpenStack Networking にセキュリティグループをプロキシする為に、以下の設定値を用いて下さい。

- firewall_driver : nova-compute が自身で iptables ベースのフィルタリングを実行しないよう、'nova.virt.firewall.NoopFirewallDriver' に設定しなければなりません。

- security_group_api : 全てのセキュリティグループ要求が OpenStack Networking サービスを経由するよう、'neutron' に設定しなければなりません。

セキュリティグループとセキュリティグループルールは、管理者とテナントが仮想インターフェースポートの通過を許可する通信のタイプと通信方向（内向き／外向き）を指定できるようにしています。セキュリティグループはセキュリティグループルールの入れ物です。OpenStack Networking 中で仮想インターフェースポートが作成された場合、ポートはセキュリティグループに紐付けられます。セキュリティグループが指定されない場合、ポートは「default」セキュリティグループに紐付けられます。デフォルトでは、このグループは内向きの通信を全てドロップし、外向きの通信を全て許可します。挙動を変える為に、このグループにルールを追加する事が出来ます。

OpenStack Compute のセキュリティグループ API を使用する場合、セキュリティグループは１インスタンス上の全仮想インターフェースポートに適用されます。この理由は、OpenStack Compute のセキュリティグループ API がインスタンスベースであり、OpenStack Networking のような仮想インターフェースポートベースではないからです。

クォータ

クォータは、テナントに対して利用可能なネットワークリソース数を制限する機能を提供します。全てのテナントに対してデフォルトのクォータを強制する事が出来ます。

```
/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 はまた、クォータ拡張 API 経由で、テナント単位のクォータをサポートしています。テナント単位クォータを有効にするためには、neutron.conf 中の quota_driver を設定する必要があります。

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

第28章 ケーススタディ: Networking

このケーススタディでは、アリスとボブがどのようにユーザに対してネットワーク提供を扱うかを議論します。

アリスのプライベートクラウド

アリスのクラウドの主目的は、既存の認証サービスとセキュリティリソースを用いたインテグレーションです。このプライベートクラウドのキーとなる設計パラメータは、テナント・ネットワーク・ワークロードタイプの限定されたスコープです。この環境は、どの利用可能なネットワークリソースがテナントから利用可能であり、どの様々なデフォルトクォータとセキュリティポリシーが利用可能かを制限するために設計される可能性があります。ネットワークポリシーエンジンは、ネットワークリソースの作成と変更を制限する為に修正される可能性があります。この環境では、アリスはインスタンス単位のセキュリティグループポリシーの適用における nova-network か、Neutron のポートベースのセキュリティグループポリシーの適用のいずれを希望するかも知れません。この環境における L2 アイソレーションは VLAN タギングを用います。VLAN タグの利用は、物理インフラの既存の機能やツールの利用によるテナントトラフィックの素晴らしい可視性が得られます。

ボブのパブリッククラウド

ボブの主要なビジネス目的は彼の顧客に先進的なネットワークサービスを提供する事です。ボブの顧客はマルチタイアのアプリケーションスタックをデプロイしようとしています。マルチタイアのアプリケーションは、既存の商用アプリケーションや新しくデプロイされるアプリケーションのいずれかです。ボブのパブリッククラウドはマルチテナントの商用サービスである為、この環境の L2 アイソレーションに使用する選択肢は、オーバレイネットワークです。ボブのクラウドの他の側面は、必要に応じて顧客が利用可能なネットワークサービスをプロビジョンできるセルフサービス指向です。これらのネットワークサービスは、L2 ネットワーク、L3 ルーティング、ネットワークACL、NAT を含みます。It is important that per-tenant quota's be implemented in this environment.

OpenStack Networking 利用の追加的な利点は、新しい先進的なネットワークサービスが利用可能になった場合、これらの新しい機能をエンドユーザに簡単に提供できる事です。

第29章 メッセージキューアーキテクチャー

メッセージキューイングサービスは、OpenStack 内におけるプロセス間通信を担います。OpenStack は次のキューイングサービスをサポートしています。

- RabbitMQ
- Qpid
- ZeroMQ、または、0MQ

RabbitMQ と Qpid は両方とも、Advanced Message Queuing Protocol (AMQP) フレームワークであり、ピアツーピア通信にメッセージキューを提供する仕組みです。 キューの実装は通常、キューサーバのプールを集中型か分散型で展開します。 ZeroMQ はピア間の通信に直接 TCP ソケットを使うところが異なっています。

メッセージキューは、OpenStack 内における指揮系統の機能を担います。一度キューへのアクセスが許可されると、その後の認証チェックは行われません。キューを使用するサービスがメッセージペイロード内のコンテキストとトークンのチェックを行います。 とはいえ、トークンの期限切れには注意を払う必要があります。これは、トークンが潜在的に再発行可能であり、インフラストラクチャ内の他のサービスを許可する可能性があるためです。

OpenStack は、メッセージへの署名のようなメッセージレベルのコンフィデンスはサポートしていません。そのため、メッセージの通信路そのものがセキュア化され、かつ、キューサーバーへのアクセスの際に認証が行なわれる必要があります。 また、HA 設定の際には、キュー間の認証と暗号化も同様に実施するべきです。

ZeroMQ メッセージングでは、IPC ソケットが各マシンで使用されます。これらのソケットは管理者がセキュア化しない限り、ローカルメッセージインジェクションやスヌーピングの攻撃に脆弱な可能性があります。

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 データベースバックエンドの考慮事項

データベースサーバーの選択は OpenStack 環境のセキュリティにおける重要な考慮事項です。セキュリティの考慮事項はデータベースサーバーの選択における唯一の基準ではありませんが、このドキュメントではこれらのみを取り扱います。実際のところ、OpenStack は 2 種類のデータベース PostgreSQL と MySQL のみをサポートします。

PostgreSQL は、Kerberos 認証、オブジェクトレベルのセキュリティ、暗号化のサポートなど、数多くの望ましいセキュリティ機能を有します。PostgreSQL コミュニティは実用的なセキュリティ実践を推進するために、わかりやすいガイダンス、ドキュメント、ツールを十分に提供してきました。

MySQL は大規模なコミュニティを持ち、幅広く適用され、高可用性のオプションを提供しています。MySQL も、プラグイン認証機構の方法により高度なクライアント認証を提供する機能があります。MySQL コミュニティから派生したディストリビューションは、考慮事項に対する多くのオプションを提供しています。セキュリティの考え方やディストリビューションに提供されるサポートレベルの評価に基づいて、特定の MySQL ディストリビューションを選択することが重要です。

データベースバックエンドのセキュリティ参考資料

MySQL や PostgreSQL を導入する人は、既存のセキュリティガイダンスを参照することが推奨されます。いくつかの参考資料を以下に一覧化します。

MySQL:

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

PostgreSQL:

- OWASP PostgreSQL Hardening
- Total security in a PostgreSQL database

第33章 データベースアクセス制御

それぞれの OpenStack コアサービス（Compute, Identity, Networking, Block Storage）は、状態や設定に関する情報をデータベースに保存します。本章では、データベースが現在 OpenStack でどのように使用されているのかを議論します。セキュリティの考慮事項、データベースバックエンドの選択によるセキュリティへの影響についても説明します。

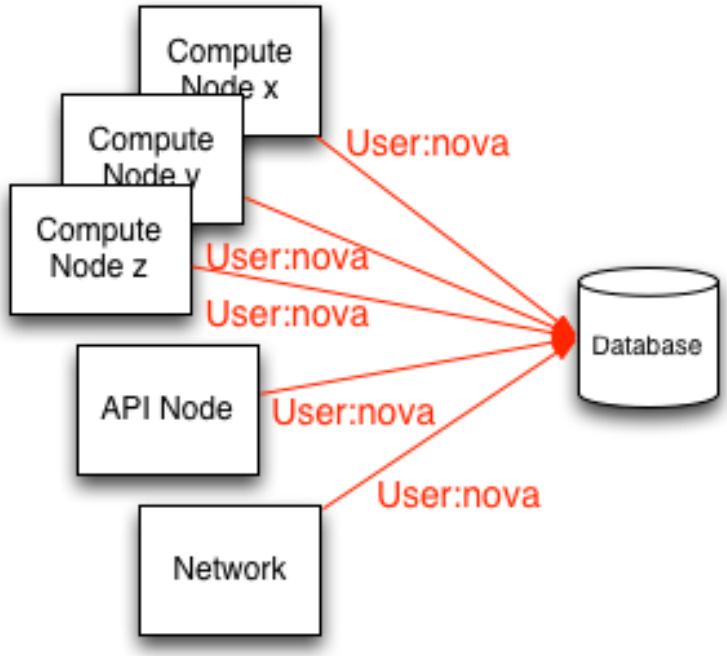
OpenStack データベースアクセスモデル

OpenStack プロジェクトの中にあるすべてのサービスは単一のデータベースにアクセスします。データベースへのテーブルの作成や行単位のアクセス制限に関する明確なポリシーは今のところありません。

OpenStack には、データベース操作の詳細な制御に関する一般的な決まりがありません。アクセス権と権限は単にノードがデータベースにアクセスするかしないかに基づいて与えられます。このシナリオでは、データベースにアクセスするノードは、DROP、INSERT、UPDATE 関数の完全な権限を持っているでしょう。

精細なアクセス制御

OpenStack の各サービスとそれらのプロセスはデフォルトで、共有クレデンシャルを使用してデータベースにアクセスします。これにより、データベース操作の監査および、サービスとそのプロセスからデータベースへのアクセス権の剥奪が特に難しくなります。

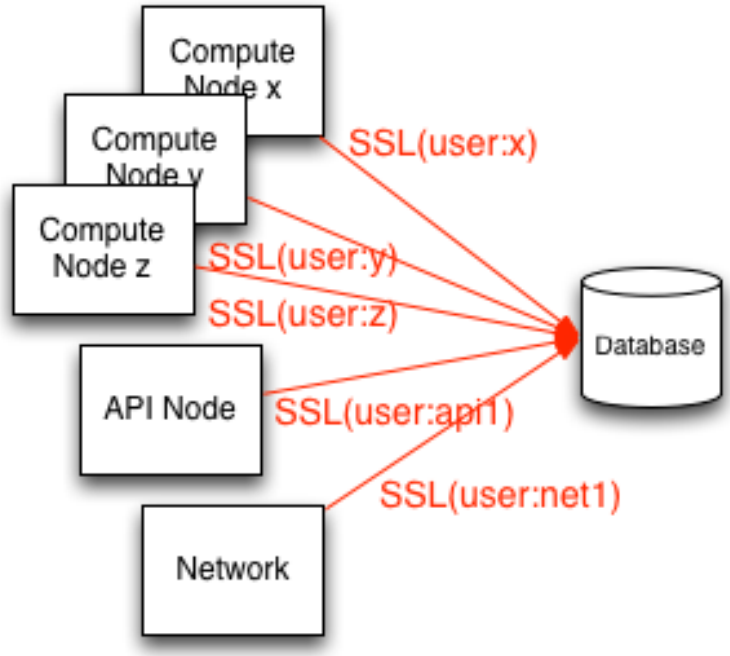


Nova Conductor

コンピュートノードは、プロジェクトのインスタンスをホストするため、OpenStack で最も信頼できないサービスです。nova-conductor サービスは、コンピュートノードとデータベースの中継役として動作する、データベースプロキシとして処理するために導入されました。その結果について本章で後ほど議論します。

以下の事項を強く推奨します。

- すべてのデータベース通信の管理ネットワークへの分離
- SSL を使用したセキュア通信
- OpenStack サービスのエンドポイントごとに一意なデータベースユーザーアカウントの作成（下図）



データベースの認証とアクセス制御

データベースにアクセスする辺りにリスクがあるため、データベースにアクセスする必要があるノードごとに一意なデータベースユーザーアカウントを作成することを強く推奨します。この機能を実行することにより、コンプライアンスを保証するため、またはノードのセキュリティ被害にあった際に分析および監査をより良くできます。また、検知した際に被害にあったノードからデータベースへのアクセス権を削除することにより、被害にあったホストを分離できます。サービスのエンドポイントのデータベースユーザーアカウントごとにこれらを作成するとき、これらに SSL を要求するよう確実に設定することに注意してください。代わりに、セキュリティを向上させるために、データベースアカウントがユーザー名とパスワードに加えて X.509 証明書認証を使用するよう設定することを推奨します。

権限

データベースの作成と削除、ユーザーアカウントの作成、ユーザーの権限の更新に関する完全な権限を持つ、別々のデータベース管理者 (DBA) アカウントが作成され、保護されるべきです。これは、不注意な設定ミ

スを防ぎ、リスクを減らし、被害の範囲を小さくする、責任の分離を実現する簡単な方法です。

データベースユーザーアカウントは OpenStack サービスのために作成され、ノードがメンバーであるサービスに関連するデータベースだけに制限された権限を持つ各ノードのために作成されます。

SSL 通信利用のための必須ユーザーアカウント

設定例 #1: (MySQL)

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

設定例 #2: (PostgreSQL)

pg_hba.conf において:

```
hostssl dbname compute01 hostname md5
```

このコマンドは SSL 経由で通信する機能を追加するのみであり、排他的ではないことに注意してください。SSL を唯一のアクセス方法にするために、暗号化されていない通信を許可するかもしれない他のアクセス方法は無効化されるべきです。

「md5」パラメーターは認証方式をハッシュ化パスワードとして定義します。以下のセクションでセキュアな認証例を提供します。

X.509 証明書を用いた認証

認証に X.509 クライアント証明書を要求することにより、セキュリティを向上させられるかもしれません。この方法でデータベースに認証することにより、データベースに接続しているクライアントの ID 確認をより強力にでき、通信が確実に暗号化されます。

設定例 #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';
```

設定例 #2: (PostgreSQL)

```
hostssl dbname compute01 hostname cert
```

OpenStack サービスのデータベース設定

お使いのデータベースサーバーが認証に X.509 証明書を要求するよう設定している場合、データベースバックエンドのために適切な SQLAlchemy クエリーパラメーターを指定する必要があります。これらのパラメーターは初期接続文字列に用いる証明書、秘密鍵、認証局の情報を指定します。

MySQL への X.509 証明書認証の :sql_connection 文字列の例:

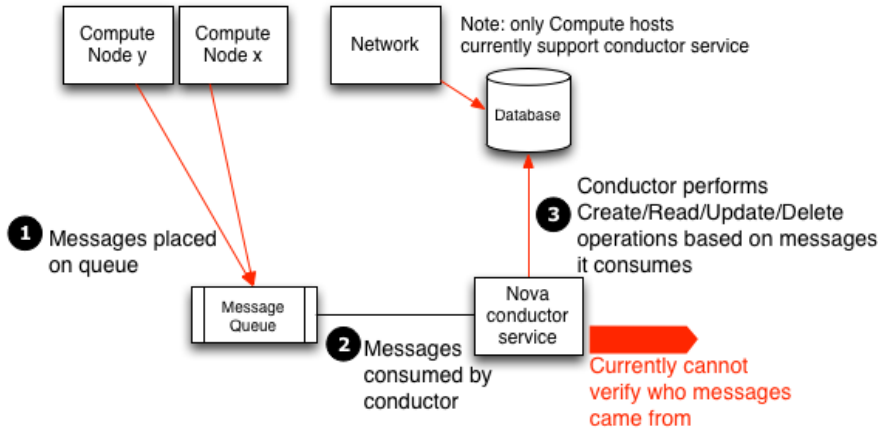
```
sql_connection = mysql://compute01:password@localhost/nova?
charset=utf8&ssl_ca=/etc/mysql/cacert.pem&ssl_cert=/etc/mysql/server-cert.
pem&ssl_key=/etc/mysql/server-key.pem
```

Nova Conductor

OpenStack Compute は nova-conductor というサブサービスを提供します。これは、nova-conductor と直接接する nova コンピュートノードがデータ永続性の要求を満たすことを主目的として、それらがデータベースと直接通信する代わりにデータベース接続を中継します。

Nova-conductor は RPC 経由でリクエストを受信します。そして、データベース、テーブル、データへの精細なアクセス権なしでサービスを呼び出す動作を実行します。Nova-conductor は本質的にコンピュートノードがデータベースに直接アクセスすることを抽象化します。

この抽象化は、サービスがパラメーター、ストアドプロシージャーのようなものを用いたメソッドの実行を制限し、数多くのシステムがデータベースのデータに直接アクセスしたり変更したりすることを防ぐという利点を提供します。これは、一般的なストアドプロシージャーという頻繁に批判される、データベース自体の文脈や範囲の中で、これらの手順を保存して実行することなく実現されます。



残念なことに、このソリューションはより詳細なアクセス制御とデータアクセスの監査機能を複雑にします。nova-conductor サービスは RPC 経由でリクエストを受信するため、メッセージングのセキュリティを改善する重要性を強調させます。メッセージキューにアクセスするすべてのノードは、nova-conductor により提供されるこれらの方式を実行し、データベースを効率的に変更するかもしれません。

最後に、Grizzly リリース時点では、nova-conductor が OpenStack Compute 全体で使用されないというギャップが存在することに注意してください。その設定に依存して、nova-conductor を使用しても、導入者が個々のコンピュートホストにデータベースの権限を与える必要性を避けられないかもしれません。

nova-conductor は OpenStack Compute のみに適用されるので、Telemetry (Ceilometer)、Networking、Block Storage のような他の OpenStack コンポーネントの動作のために、コンピュートホストから直接データベースにアクセスする必要があるかもしれないことに注意してください。

導入者は nova-conductor を有効化または無効化する前に両方の設定の利点とリスクを比較検討すべきです。Grizzly リリースでは nova-conductor の利用を推奨する準備ができていません。しかしながら、追加の機能が OpenStack にもたらされるので、この推奨について変更されると確信しています。

nova-conductor を無効化するために、以下の事項を（コンピュートホストの）nova.conf ファイルに記入します。

```
[conductor]
```

```
use_local = true
```

第34章 データベース通信セキュリティ

本章はデータベースとのネットワーク通信に関連する問題を取り扱います。これには、IP アドレスのバインドや SSL を用いた暗号化ネットワーク通信を含みます。

データベースサーバーの IP アドレスバインド

サービスとデータベース間の機微なデータベース通信を隔離するために、データベースサーバーが隔離された管理ネットワーク経由のみでデータベースと通信できるように設定することを強く推奨します。データベースサーバーがクライアントからの通信用のネットワークソケットをバインドするインターフェースまたは IP アドレスを制限することにより、これを実現できます。

MySQL のバインドアドレスの制限

my.cnf の場合:

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

PostgreSQL のバインドアドレスの制限

postgresql.conf の場合:

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

データベース通信

データベース通信を管理ネットワークに制限することに加えて、クラウド管理者がそれらのデータベースのバックエンドに SSL を要求するよ

うに設定することを強く推奨します。データベースのクライアント接続に SSL を使用することにより、改ざんや盗聴から通信を保護できます。次のセクションで議論するように、SSL を使用することにより、データベースのユーザー認証に X.509 証明書（一般的に PKI として参照されます）を使用するフレームワークも提供できます。以下は、2 つの有名なデータベースバックエンド MySQL と PostgreSQL に SSL を典型的に設定する方法について示します。

注記

注: 証明書と鍵ファイルをインストールするとき、ファイルのパーミッションが制限されていることを確認します。たとえば、chmod 0600 を実行すると、データベースサーバー上の他のプロセスやユーザーによる権限のないアクセスを防ぐために、所有者がデータベースデーモンのユーザーに制限されます。

MySQL SSL 設定

以下の行をシステム全体の MySQL 設定ファイルに追加する必要があります。

my.cnf の場合:

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

オプションとして、暗号化通信に使用される SSL 暗号を制限したい場合、暗号の一覧と暗号文字列を設定するための構文は http://www.openssl.org/docs/apps/ciphers.html を参照してください。

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

PostgreSQL SSL 設定

以下の行をシステム全体の PostgreSQL 設定ファイル postgresql.conf に追加する必要があります。

```
ssl = true
```

オプションとして、暗号化通信に使用される SSL 暗号を制限したい場合、暗号の一覧と暗号文字列を設定するための構文は http://www.openssl.org/docs/apps/ciphers.html を参照してください。

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

サーバー証明書、鍵、認証局（CA）のファイルを $PGDATA ディレクトリの以下のファイルに置く必要があります。

- $PGDATA/server.crt - サーバー証明書
- $PGDATA/server.key - server.crt に対応する秘密鍵
- $PGDATA/root.crt - 信頼された認証局
- $PGDATA/root.crl - 証明書失効リスト

第35章 ケーススタディ: データベース

このケーススタディでは、アリスとボブがどのようにデータベースを選択し、それぞれのプライベートクラウドとパブリッククラウド用に設定するのかについて議論します。

アリスのプライベートクラウド

アリスの組織は高可用性に関心があります。そのため、データベースに MySQL を使用することにしました。彼女はさらに、管理ネットワークにデータベースを配置し、アクセスを確実にセキュアにするために、サービス間の相互認証とともに SSL を使用します。データベースの外部アクセスはなく、データベースとそのアクセスエンドポイントに、組織の自己署名ルート証明書で署名した証明書を使用します。アリスは各データベースユーザーに対して別々のユーザーアカウントを作成し、認証のためにパスワードと X.509 証明書の両方を使用するようデータベースを設定します。高精細なアクセス制御ポリシーと監査をサポートしたいので、nova-conductor サブサービスを使用しないことにします。

ボブのパブリッククラウド

ボブはプロジェクトのデータの確実な分離に関心があります。そのため、彼はより強力なセキュリティ機能が知られている Postgres データベースを使用することにしました。データベースは管理ネットワークに置かれ、サービス間の相互認証とともに SSL を使用します。データベースは管理ネットワークにあるので、組織の自己署名ルート証明書で署名した証明書を使用します。ボブは各データベースユーザーに対して別々のユーザーアカウントを作成し、認証のためにパスワードと X.509 証明書の両方を使用するようデータベースを設定します。高精細なアクセス制御をしたいので、nova-conductor サブサービスを使用しないことにします。

第36章 データプライバシ関連

OpenStack はマルチテナンシーをサポートするよう設計されており、これらのテナントには、まず間違いなく異なるデータ要件があるでしょう。クラウド構築者とオペレータとして、あなたは自身の OpenStack 環境が様々なデータプライバシー関連と規制を扱える事を確認する必要があります。OpenStack 実装に関連するので、本章ではデータプライバシーにまつわる以下のトピックを扱います。

- データの所在
- データの処分

データの所在

データのプライバシーと分割は、ここ数年クラウド採用の最初の障壁としてずっと言及されてきました。クラウド中でデータを所有するのは誰か、このデータの管理人としてクラウドオペレータは結局信用できるのか否かという事は、これまで重要な問題でした。

多数の OpenStack サービス群は、テナントやテナント情報の所在に属するデータとメタデータを管理します。

OpenStack クラウドに保存されたテナントデータは以下の項目を含まれます：

- Swift オブジェクト
- Compute のインスタンスの一時的ファイルシステムストレージ
- Compute インスタンスのメモリ
- Cinder のボリュームデータ
- Comptue アクセス用パブリックキー
- Glance 中の仮想マシンイメージ
- マシンのスナップショット
- OpenStack Compute の設定用ドライブ拡張に渡されたデータ

以下の不完全な一覧を含む、OpenStack クラウドが保存したメタデータ:

- 組織名
- ユーザの「実名」
- 実行中のインスタンスのサイズ、バケット、オブジェクト、ボリューム、その他クォータ関連の項目
- 実行中のインスタンス又は保存されたデータの経過時間
- ユーザの IP アドレス
- Compute イメージ作成用に内部で生成されたプライベートキー

データの処分

OpenStack オペレータは、ある一定レベルのテナントデータ破棄保証が提供できるよう努力しなければんりません。ベストプラクティスは、クラウドシステムメディア（デジタル・非デジタル）を破棄、組織コントロール外へのリリース、再利用の為の開放より前にオペレータがメディアをクリアする事を推奨しています。メディアのクリア方法は、特定のセキュリティドメインと情報のデリケートさが与えられた、適切なレベルの強度と完全性を実装すべきです。

> 「データのサニタイズは、情報が取得あるいは再構築できない事の合理的な保証が得られるよう、システム媒体から情報を削除する為に使用されるプロセスです。サニタイズ技術（媒体の情報のクリア、破棄、破壊を含む）は、こうした媒体が再利用・譲渡・破棄された際に、組織の情報が閲覧権限のない個人に開示される事を防ぎます。」 [NIST Special Publication 800-53 Revision 3]

NIST が採用した汎用のデータ破棄とサニタイズのガイドラインは、セキュリティ制御を推奨しています。クラウドオペレータは以下のことをすべきです。

1. 媒体サニタイズと破棄行為の追跡・文書化・検証を行うこと。
2. 適切なパフォーマンスを検証する為、サニタイズ設備と過程の評価を行うこと。
3. 持ち運び可能なリムーバルストレージデバイスをクラウドインフラに接続する前にサニタイズすること。

4. サニタイズできないクラウドシステム媒体を破壊すること。

OpenStack デプロイでは、以下の事も実施する必要があるでしょう。

- 安全なデータの消去
- インスタンスメモリの消去
- ブロックストレージボリュームデータ
- Compute インスタンスの一時ストレージ
- 物理サーバのサニタイズ

安全に消去されなかったデータ

OpenStack 中でいくつかのデータは削除されるかも知れませんが、上記で触れた NIST 標準の文脈における安全な消去ではありません。これは一般に、データベースに保存された上記で定義したメタデータと情報の大半又は全てに当てはまります。これは、データベースとシステム設定のどちらか又は両方で、自動バキュームと定期的な空き領域のクリアを実施する事で解決する事ができるかも知れません。

インスタンスメモリの消去

様々なハイパーバイザの特色はインスタンスメモリの扱いにあります。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. この挙動は OpenStack Compute で定義されておらず、ハイパーバイザがインスタンス作成時または削除時、あるいはその両方において、ベストエフォートでメモリのクリンアップを行うだろうと一般に考えられています。

Xen は、専用のメモリ範囲をインスタンスに明確に割り当て、インスタンス（又は Xen の用語でドメイン）破棄時にそのデータをクリンアップします。KVM はより大いに Linux のページ管理に依存しています。 KVM のページングに関する複雑なルールセットは、KVM の文書で定義されています。

Xen のメモリバルーン機能の使用は情報漏えいの結果になりかねないという事への注意は重要です。

これらや他のハイパーバイザでは、ハイパーバイザ毎のドキュメントを参照すると良いでしょう。

Cinder のボリュームデータ

OpenStack Block Storage のプラグインは様々な方法でデータの保存を行います。多くのプラグインはベンダー又はストレージ技術に特化していますが、その他は LVM や ZFS といったファイルシステム辺りのより手作りのソリューションです。安全にデータを破壊する方法はプラグイン毎、ベンダーのソリューション毎、ファイルシステム毎に異なるでしょう。

ZFS のようないくつかのバックエンドは、データの漏洩を防ぐために copy-on-write に対応しています。この場合、まだ書き込まれていないブロックからの読み込みは常にゼロを返します。LVM のような他のバックエンドでは copy-on-write を標準でサポートしておらず、よって Cinder プラグインが以前に書き込まれたブロックをユーザがアクセスする前に上書きする役割を担います。あなたが選択したボリュームバックエンドが提供する機能をレビューし、これらの機能が提供しない事についての回避策が利用できるかを調べる事は重要です。

最後に、これは OpenStack の機能ではありませんが、ベンダーと開発者がボリュームの暗号化機能をサポートするか、あるいは追加可能であるかも知れません。この場合、データの破壊は単にキーを破棄するだけです。

Compute インスタンスの一時ストレージ

一時ストレージの作成・削除は選択したハイパーバイザや OpenStack Compute プラグインに依存するでしょう。

compute 用の libvirt プラグインは、ファイルシステム又は LVM 上の一時ストレージを直接管理出来ます。ファイルシステムストレージは一般にデータを削除する際に上書きはしませんが、ユーザに対して汚れたエクステンドが用意されないという保証があります。

ブロックデバイスベースである LVM をバックエンドにした一時ストレージを使用する場合、OpenStack Compute は情報漏えいを防ぐために、安全にブロックを削除する必要があります。これらには、過去において、不適切な一時ブロックストレージデバイスの削除に関連する情報漏洩の脆弱性がありました。

データが含まれたエクステンドがユーザに用意されないので、一時ブロックストレージデバイス用としてファイルシステムストレージは LVM より安全なソリューションです。しかしながら、ユーザデータが破壊されない事を覚えておく事は重要であり、このためバックエンドのファイルシステムの暗号化が提案されています。

物理サーバのサニタイズ

Nova の物理サーバドライバは開発中だったのですが、Ironicと呼ばれる独立したプロジェクトに移管される事になりました。この文書の執筆時点では、Ironic には物理ハードウェア上にあるテナントデータのサニタイズ機能はまだありません。

加えて、物理マシンのテナントでは、システムファームウェアの修正が可能です。##link:Management/Node Bootstrapping## で説明されている TPM 技術は、許可されていないファームウェアの変更を検知する解決策を提供します。

第37章 データ暗号化

このオプションは、データをディスクに保存する箇所やデータをネットワーク経由で転送する箇所でテナントデータを暗号化する実装者用です。ユーザが自分自身のデータをプロバイダに送信する前にデータを暗号化するという一般的な推奨の上またはその先にあるものです。

テナントの為のデータ暗号化の重要性は、攻撃者がテナントデータにアクセスできる事をプロバイダが想定するリスクに広く関係しています。政府での要件があるかも知れませんし、（ポリシー単位の要件と同様）パブリッククラウド提供者用の随意契約に関しては、随意契約の中、あるいは判例法の中でさえ要求されるかも知れません。テナント暗号化ポリシーを選択する前に、リスク分析と法務コンサルの忠告を受ける事をお勧めします。

暗号化の単位は、好ましい方から順にインスタンス単位又はオブジェクト単位、プロジェクト単位、テナント単位、ホスト単位、クラウド集合単位です。この推奨順は、実装の複雑さと困難さの順序の逆です。現在、いくつかのプロジェクトでは、テナント単位ですら粒度の荒い暗号化の実装が困難又は不可能です。実装者がテナントデータの暗号化を最善策とする事をお勧めします。

時々、データ暗号化は単に暗号鍵を捨てるという事による、信頼できるテナントやインスタンス単位のデータ削除可能性と明確に関係があります。そうするよう記述すべきですし、信頼できる安全な方法でこれらの鍵を破壊する事が従来になります。

ユーザ用のデータ暗号化をする機会は現存します。

- Object Storage オブジェクト
- Block Storage ボリューム & インスタンスの一時ファイルシステム
- ネットワークデータ

Object Storage オブジェクト

Object Storage 中のオブジェクトの暗号化の可能性は、現時点ではノード単位のディスクレベル暗号化に限定されています。しかしながら、オ

ブジェクト単位の暗号化用のサードパーティ拡張やモジュールが存在します。これらのモジュールはアップストリームに提案されていますが、この文書を書いている時点では公式に認可されていません。下記はそれらの幾つかへのポインタです。

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

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

Block Storage ボリューム & インスタンスの一時ファイルシステム

暗号化ボリュームの可否は選択したサービスバックエンドに依存します。いくつかのバックエンドは暗号化を全くサポートしないかも知れません。

Block Storage と Compute は両方、LVM ベースのストレージをサポートしているので、両システムに簡単に適用可能な例を提供します。LVM を用いたデプロイでは、暗号化はベースの物理ボリュームに対して実施できます。暗号化ブロックデバイスは、pvcreate を使用して復号化したブロックデバイスの上に作成した LVM 物理ボリューム（PV）を用いて、標準の Linux ツールを使用して作成する事ができます。それから、vgcreate 又は vgmodify ツールを使用して、暗号化した物理ボリュームを LVM のボリュームグループ（VG）に追加できます。

Havana リリース向けの 1 機能が、ディスクに書き込まれる前の VM データの暗号化を提供しています。これは、ストレージデバイス上でもデータのプライバシーが管理される事を可能にします。このアイデアは自己暗号化ドライブが機能する方法と同様です。この機能は、VM には通常のブロックストレージデバイスとして見えますが、仮想化ホストではディスクにデータが書き込まれる前にデータが暗号化されます。ブロックサーバは、特別な処理がスナップショットやライブマイグレーションといった Block Storage の機能に向けて要求される事を除いて、暗号化されていないブロックを読み書きする場合と全く同様に処理が行われます。この機能は独立した鍵管理を使用する事に注意して下さい。

ネットワークデータ

compute のテナントデータは IPSec 又は他のトンネルで暗号化できます。OpenStack での共通または標準の機能ではありませんが、やる気と興味がある実装者に 1 つの選択肢が利用できます。

Block Storage は、マウント可能なボリュームの提供に向けた様々な機構をサポートします。Block Storage の各バックエンドドライバ用に推奨を指定する事はこのガイドの範囲外です。性能の為に、多くのストレージプロトコルは暗号化されていません。iSCSI のような幾つかのプロトコルは、認証と暗号化セッションを提供できます。これらの機能を有効にする事を推奨します。

第38章 鍵管理

頻繁に触れられるテナントデータのプライバシーとクラウドプロバイダーの法的責任の限度についての懸念に対処するために、OpenStack コミュニティはデーターの暗号化を様々な個所へ適用することに興味を持っています。エンドユーザーがクラウドにデータをセーブする前にそれらを暗号化することは比較的に簡単で、メディアファイル、データベースアーカイブなどテナントオブジェクトに実行可能な方法です。しかし、クライアント側の暗号化が仮想マシンのイメージを使用する場合、ブロックストレージなどクライアントの介入では、データの更なる利用のために解除する鍵を提示する形式が必要です。しかし、クライアント側の暗号化が仮想マシンのイメージを使用する場合、ブロックストレージなどクライアントの介入では、データの更なる利用のために解除する鍵を提示する形式が必要です。シームレスにデータを保護し、クライアントの鍵を管理し対話的に鍵を提供することで負担をかけることなく、それがアクセスできるようにするには、OpenStack 内に鍵管理サービスを求められます。OpenStack の一環として、暗号化と鍵管理サービスの提供は、データ保存セキュリティ導入を容易にし、クラウド･プロバイダーの法的責任を制限する追加の利点と、プライバシーとデータの誤使用に関する顧客の懸念に対処しています。プロバイダの法的責任は、マルチテナントのパブリッククラウドで誤った調査によってテナントデータを引き渡す事が懸念されています。

鍵管理は、まだ開発の初期段階で、OpenStack の正式コンポーネントになる過程の途中です。詳細は、https://github.com/cloudkeep/barbican/wiki/_pagesを参照してください。

鍵管理は、鍵の作成と安全な保存(サービスマスター鍵付き)をサポートしなければなりません。設計上の問題として現状まだ議論されているものとして、いくつの相互運用性プロトコル(KMIP)をサポートするか、鍵の形式、認証管理があります。鍵管理は、第三者のハードウェアセキュリティモジュール(HSM)を必要とする配備を容易化するためにプラグイン可能になります。

OpenStack のブロックストレージ Cinder は、ボリュームの暗号化を提供するために鍵管理との統合を検討した最初のサービスです。

参考資料：

- Barbican

- KMIP

第39章 ケーススタディ: テナントデータ

アリスとボブの話に戻ると、このセクションでは、彼らの特定のテナントのデータのプライバシー要件についてより詳細に説明します。具体的には、アリスとボブの両者がテナントのデータ、データの破壊、データの暗号化をどのように対処するかを見てみます。

アリスのプライベートクラウド

アリスのケーススタディで説明したように、データ保護は非常に重要です。アリスは、あるテナントのデータの情報漏洩が、他のテナントデータの損害を引き起こさないように、保証することが必要です。アリスはまた、データ破壊の文書化を必要とする強い規制上の要件を持っています。アリスは、以下の方法でこれを提供します：

- プログラムやプロジェクトが終了する際に、好ましくないテナントデータを削除するための手順を確立すること
- CMDBのチケット発行を使用して、顧客データとメタデータの両方の破壊を追跡する
- ボリュームストレージ
- 物理サーバーの問題
- 安全な一時ディスクを提供するために、アリスは暗号化ファイルシステム上に qcow2 のファイルを実装しています。

ボブのパブリッククラウド

ボブのケーススタディの最初で説明したように、テナントのプライバシーは非常に重要です。ボブはインフラレイヤーで相互にテナントを分離する要件およびアクションに加えて、ボブはまたテナントデータのプライバシーを保する必要があります。 ボブは以下を用いて、これを提供します：

- 顧客がデータを量産する時に、好ましくない顧客データを削除するための手順を確立する

- CMDBのチケット発行を使用して、顧客データとメタデータの両方の破壊を追跡します。
- ボリュームストレージ
- 物理サーバーの問題
- 安全な一時ディスクを提供するために、ボブは暗号化ファイルシステム上に qcow2 のファイルを実装しています。

第40章 ハイパーバイザーの選択

仮想化はクラウドを構築できるようにする柔軟性や他の利点を提供します。しかしながら、仮想化スタックは、ハイパーバイザーへの攻撃に関連するリスクを減らすために、適切にセキュア化する必要もあります。つまり、仮想化スタックがインスタンスやゲスト仮想マシン間を分離できても、分離が完全ではない状況があります。仮想化スタックを理解して選択すること、本章に書かれているベストプラクティスに従うことは、クラウドセキュリティの階層的なアプローチに含めることができます。最後に、パブリッククラウドにおける顧客間、プライベートクラウドにおける部門間、ハイブリッドクラウドにおける両者間で、マルチテナントを前提に提供するために、仮想化スタックのセキュア化は必須です。

本章では、ハイパーバイザーの選択について説明します。さらに以降の章では、仮想スタックを安全に保つために必要な基礎情報を説明します。

OpenStack におけるハイパーバイザー

OpenStack がプライベートデータセンターかパブリッククラウドサービスのどちらにデプロイされたかに関わらず、基板となる仮想化技術はスケーラビリティ、リソース効率、稼働時間の範囲において商用レベルの機能を提供します。多くの OpenStack 対応のハイパーバイザー技術全般でこのような高いメリットが利用可能である一方、（特に様々な OpenStack 環境で固有のセキュリティ取り扱い方針を考慮する際）各ハイパーバイザーのセキュリティアーキテクチャや機能においては明確な違いがあります。アプリケーション群が単一の Infrastructure as a Service（IaaS）プラットフォームに統合される為、ハイパーバイザーレベルでのインスタンスの隔離は特に重要となります。セキュアな隔離の要件は商用・政府・軍事コミュニティ全体に当てはまります。

OpenStack のフレームワークの中で、クラウド環境を最適化するために、いくつものハイパーバイザーおよび対応する OpenStack プラグインから選択できます。OpenStack セキュリティガイドの観点では、ハイパーバイザーはセキュリティに必須となる機能セットに関連するため、ハイパーバイザーの選択における考慮事項について注目します。しかしながら、これらの考慮事項は特定のハイパーバイザーの得失について徹底的に調査したことを意味するわけではありません。NIST は Special

Publication 800-125, "Guide to Security for Full Virtualization Technologies" でさらなるガイドラインを提供しています。

選択基準

ハイパーバイザーの選択において、セキュリティを保証するために考慮すべき重要な要因がいくつかあります。特に下記の面に注目します。

- チーム習熟度
- 製品やプロジェクトの成熟度
- 認証、証明
- 追加のセキュリティ機能
- ハードウェア対ベアメタル
- ハードウェア関連
- コモンクライテリア (Common Criteria)

加えて、OpenStack デプロイ用のハイパーバイザーの選択時、以下のセキュリティ関連の条件の評価を推奨します。

- ハイパーバイザーはCommon Criteria認定を取得していますか？取得している場合はどのレベルですか？
- 採用している暗号化技術は第三者によって認定されていますか？

チームのノウハウ

多分、ハイパーバイザー選択における一番重要な観点はある特定のハイパーバイザープラットフォームの管理と保守におけるあなたのスタッフのノウハウです。あなたのチームが与えられた製品、その設定、クセに慣れていればいるほど、設定ミスは少なくなります。加えて、あなたのスタッフが与えられたハイパーバイザーについて組織を横断してノウハウを広めていけば、あなたのシステムの可用性は向上し、職務分掌の開発が可能になり、チームメンバーが対応できない場合での問題を軽減します。

製品やプロジェクトの成熟度

ハイパーバイザー製品またはプロジェクトの成熟度もセキュリティ上重要です。製品の成熟度はクラウドを配備してから大きな影響が現れます。セキュリティガイドでは、下記の面に注目します。

- ノウハウの入手先
- 活発な開発者とユーザーのコミュニティ
- タイムラインとアップデートの入手先
- インシデントレスポンス

ハイパーバイザーの完成度の最大の指標の１つに、それを取り巻くコミュニティのサイズと活気があります。これはセキュリティに関するので、コミュニティの質はあなたが追加のクラウドオペレーターを必要とする、利用可能なノウハウに影響します。これはまた、ハイパーバイザーがいかに広く開発されているかの印でもあり、同様に、リファレンスアーキテクチャやベストプラクティスの戦闘準備につながるのです。

さらに、コミュニティが KVM や Xen のようなオープンソースのハイパーバイザーを取り巻くので、その質はバグ修正やセキュリティ更新の適時性に直接的な影響があります。商用ハイパーバイザーとオープンソースのものを調査するとき、リリース間隔やサポートサイクルだけではなく、バグやセキュリティ問題のアナウンスから、パッチや対応までの時間間隔を調査したいでしょう。最後に、OpenStack Compute のサポート能力は、お使いのハイパーバイザーにより異なります。ハイパーバイザーによりサポートされる OpenStack Compute の機能は、OpenStack Hypervisor Support Matrix を参照してください。

証明書

ハイパーバイザーを選択する際にもう１つ考慮すべき点は、様々な公式の認証や証明書が利用可能かという事です。あなたの特定の組織の要件ではないかも知れませんが、これらの認証や証明書は、成熟度、商利用可能、特定のハイパーバイザーが目標としてきたテストの徹底さを物語ります。

コモンクライテリア（Common Criteria）

共通の条件は国際的に標準化されたソフトウェア評価プロセスです。これは、宣伝目的でソフトウェア技術の実行を検証する為に政府や企業が使用します。政府部門では、NSTISSP No. 11 のみ政府機関にコモンクライテリア認証（2002年7月に登場したポリシー）を受けたソフトウェアの調達権限を与えます。特に、Opentack はコモンクライテリア認証を受けておらず、多くの入手可能なハイパーバイザーは受けている事に注意すべきでしょう。

Common Criteria のプロセスは、技術的な機能の評価に加えて、技術がどのように開発されているのかを評価します。

- どのようにしてソースコード管理が行われるのか？
- どのようにしてユーザがビルドシステムへのアクセスを許可されるのか？
- 技術は配布前に暗号署名されるのか？

KVM ハイパーバイザーはアメリカ政府から Common Criteria 認証された商用ディストリビューションです。インスタンス分離を強制するための基礎的な技術を提供し、仮想マシンの実行環境を分離できることが検証されました。仮想マシンの分離に加えて、KVM は次のとおり Common Criteria 認証されています。

> ”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”（システム固有の分離機構を仮想マシンのリソースに提供する。この分離により、各仮想マシン用に実行される仮想および擬似デバイスに対して使用される大規模なソフトウェアコンポーネントが、お互いに干渉しないことを保証する。仮想マシンの管理フレームワークは、管理者に対して SELinux のマルチカテゴリ設定を透過的に設定する。）

Red Hat、Microsoft、VMWare のような多くのハイパーバイザーベンダーは、Common Criteria 認証を取得していますが、基礎となる機能セットは異なります。以下の要件を最低限確実に満たすために、ベンダーの請求内容を評価することを推奨します。

IDと認証	pluggable authentication modules (PAM) を使用した識別と認証はユーザーパスワードに基づいています。使用されるパスワードの質は設定オプションにより強制できます。
監査	システムは、個々のシステムコールを含む大多数のイベントおよび信頼されたプロセスにより生成されたイベントを監査する機能を提供します。監査データは通常のファイルに ASCII 形式で収集されます。システ

	ムは、監査レコードを検索するためのプログラムを提供します。 システム管理者は、関心のあるイベントに監査を制限するために、ルールベースを定義できます。これには、特定のイベント、特定のユーザー、特定のオブジェクトやこれらすべての組み合わせに監査を制限する機能が含まれます。 監査レコードはリモート監査デーモンに転送できます。
任意アクセス制御	任意アクセス制御（DAC）は、ユーザー、グループ、その他に対する標準 UNIX パーミッションを含むアクセス制御リスト（ACL）に基づいてファイルシステムオブジェクトへのアクセスを制限します。アクセス制御機構は権限のないアクセスから IPC オブジェクトも保護します。 システムは POSIX ACL をサポートする ext4 ファイルシステムを含みます。この種類のファイルシステムにあるファイルにユーザー単位でアクセス権を定義できます。
強制アクセス制御	強制アクセス制御（MAC）は、サブジェクト（主体）とオブジェクト（対象）に割り当てられたラベルに基づいて、オブジェクトへのアクセスを制限します。機密性のラベルがプロセスとオブジェクトに自動的に付けられます。これらのラベルを使用して強制されたアクセス制御ポリシーは、BellLaPadula アクセス制御モデルから派生したものです。 SELinux カテゴリが仮想マシンとそのリソースに付けられます。仮想マシンのカテゴリがアクセスされるリソースのカテゴリと同じ場合、これらのカテゴリを使用して強制されたアクセス制御ポリシーは、仮想マシンのそのリソースへのアクセスが許可されます。 TOE は、仮想マシンへのアクセスを制御するために、非階層的なカテゴリを実装します。
ロールベースアクセス制御	ロールベースアクセス制御（RBAC）は、全権を持つシステム管理者の必要性を減らすために、役割を分割できるようにします。
オブジェクト再利用	ファイルシステムのオブジェクト、メモリ、IPC オブジェクトは、他の

	ユーザーに属するプロセスにより再利用される前に、クリアされます。
セキュリティ管理	セキュリティ的に重要なシステムパラメーターの管理が、管理ユーザーにより実行されます。root 権限（または RBAC 使用時の特定のロール）が必要となる一組のコマンドが、システム管理のために使用されます。セキュリティ関連のパラメーターは特定のファイルに保存されます。これらは、システムのアクセス制御機構により、管理ユーザー以外の権限のないアクセスに対して保護されます。
セキュア通信	システムは SSH を使用する信頼チャネルの定義をサポートします。パスワードによる認証がサポートされます。少しの暗号スイートのみが、評価された設定でそれらのプロトコルのためにサポートされます。
ストレージ暗号化	システムは dm_crypt 経由でストレージの機密性を提供するために暗号化ブロックデバイスを提供します。
TSF 保護	動作中、カーネルソフトウェアとデータがハードウェアメモリ保護機構により保護されます。カーネルのメモリとプロセスの管理コンポーネントにより、ユーザープロセスがカーネルストレージや他のプロセスのストレージにアクセスできないことが保証されます。 非カーネル TSF ソフトウェアとデータが DAC とプロセス分離機構により保護されます。評価済みの設定で、予約済みユーザー ID root は TSF 設定を定義するディレクトリとファイルを所有します。一般的に、設定ファイルやバッチジョブのキューのような、内部 TSF データを含むファイルとディレクトリも、DAC パーミッションにより読み取りから保護されます。 システム、ハードウェア、ファームウェアのコンポーネントは、権限のないアクセスから物理的に保護される必要があります。システムカーネルは、プログラムから利用できる CPU 命令ファンクション以外に、ハードウェア機構自身へのすべてのアクセスを調停します。

	さらに、スタックオーバーフロー攻撃に対する保護機構が提供されます。

暗号標準

いくつかの暗号アルゴリズムは、認証と識別、データ転送、保存データの保護のために、OpenStack の中で利用可能です。ハイパーバイザーの選択時、以下が推奨アルゴリズムで、仮想化層のサポートを確実にするための実装標準です。

アルゴリズム	鍵の長さ	想定用途	セキュリティ機能	実装標準
AES	128 ビット、192 ビット 256 ビット	暗号化 / 復号	保護されたデータ転送、保存データの保護	RFC 4253
TDES	168 ビット	暗号化 / 復号	保護されたデータ転送	RFC 4253
RSA	1024 ビット、2048 ビット 3072 ビット	認証、鍵交換	識別と認証、保護されたデータ転送	U.S. NIST FIPS PUB 186-3
DSA	L=1024,N=160 ビット	認証、鍵交換	識別と認証、保護されたデータ転送	U.S. NIST FIPS PUB 186-3
Serpent	128、196、256 ビット	暗号化 / 復号	保存データの保護	http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf
Twofish	128、196、256 ビット	暗号化 / 復号	保存データの保護	http://www.schneier.com/paper-twofish-paper.html
SHA-1	-	メッセージダイジェスト	保存データの保護、保護されたデータ転送	U.S. NIST FIPS 180-3
SHA-2(224、256、 384、512 ビット)		メッセージダイジェスト	保存データの保護、識別と認証	U.S. NIST FIPS 180-3

FIPS 140-2

アメリカでは、National Institute of Science and Technology (NIST) が Cryptographic Module Validation Program として知られるプロセスにより暗号アルゴリズムを認証します。NIST は、以下を保証する Federal Information Processing Standard 140-2 (FIPS 140-2) に適合するアルゴリズムを認証します。

> FIPS 140-2 への適合性を検証された製品は、機密情報（アメリカ）や指定情報（カナダ）の保護のために、両国（アメリカとカナダ）の連邦機関により受け入れられます。CMVP の目標は、検証済み暗号モジュールを含む物品調達で使用するために、検証済み暗号モジュール利用を推進することと、連邦機関へのセキュリティ評価基準を提供することです。

ハイパーバイザーの基礎技術の評価時、ハイパーバイザーが FIPS 140-2 に認証されているかどうかを考慮します。正式な認証は、指定された暗号アルゴリズムの実装が、アメリカ政府機関のポリシーごとに強制される FIPS 140-2 への適合性だけではなく、モジュール仕様、暗号モジュールのポートとインターフェース、ロール、サービス、認証、有限オートマトン、物理セキュリティ、運用環境、暗号鍵管理、EMI/EMC、自己テスト、設計保証、他の攻撃の緩和に対する適合性をレビューされることを意味します。

ハードウェア関連

さらに、ハイパーバイザープラットフォームの評価時、ハイパーバイザーを実行するハイパーバイザーを考慮すべきです。加えて、ハードウェアで利用可能な追加機能を評価します。また、それらの機能が OpenStack 環境の一部として選択したハイパーバイザーによりどのようにサポートされるかを考慮します。そのためにも、ハイパーバイザーはそれぞれ自身のハードウェア互換性リスト (HCL) を持つでしょう。互換性のあるハードウェアの選択時、まずどのハードウェア仮想化技術がセキュリティの観点から重要であるかを理解することが重要です。

記述	技術	説明
I/O MMU	VT-d / AMD-Vi	PCI パススルーの保護に必要です
Intel Trusted Execution Technology	Intel TXT / SEM	動的証明サービスに必要です
PCI-SIG I/O 仮想化	SR-IOV, MR-IOV, ATS	PCI Express デバイスをセキュアに共有するために必要です

ネットワーク仮想化	VT-c	ハイパーバイザーにおけるネットワーク I/O の性能を改善します

ハードウェア対ベアメタル

ハイパーバイザーの選択に関する議論をまとめるために、LXC（Linux コンテナー）やベアメタルシステムの利用と KVM のようなハイパーバイザーの利用の違いを思い起こすことが重要です。具体的には、このセキュリティガイドの焦点は、大規模にハイパーバイザーと仮想化のプラットフォームを持つことを前提にしています。しかしながら、お使いの環境がベアメタルや LXC 環境を使用する必要があれば、その環境に関する特有の違いに注意を払いたいでしょう。とくに、ノードが再配備する前にデータを適切に無害化されることをエンドユーザーに保証する必要があります。加えて、ノードを再利用する前に、ハードウェアが汚染されていたり、侵入されたりしていないことを保証する必要があります。

OpenStack はベアメタルのプロジェクトを持ちますが、ベアメタル実行の具体的なセキュリティ実装に関する議論は本書の範囲外であることに注意すべきです。

最後に、Book Sprint の時間的制約のため、実装例とアーキテクチャー例にハイパーバイザーとして KVM を使用することにしました。

注記

use of LXC in Nova に関する OpenStack Security Note があります。

追加のセキュリティ機能

ハイパーバイザー選択時に検討すべき他の事項は、特定のセキュリティ機能の利用可否です。とくに、Xen Server の XSM (Xen Security Modules)、sVirt、Intel TXT、AppArmor のような機能を利用しています。これらの機能の存在は、セキュリティプロファイルを向上するだけでなく、良い基盤を提供する役に立つでしょう。

以下の表は一般的なハイパーバイザーにおけるこれらの機能の対応状況を示します。

	KSM	XSM	sVirt	TXT	AppArmor	cGroups	MAC ポリシー
KVM	X		X	X	x	x	x

Xen		X		X			x
ESXi				X			
Hyper-V							

KSM: Kernel Samepage Merging

XSM: Xen セキュリティモジュール

xVirt: Linux ベースの仮想化向けの強制アクセス制御

TXT: Intel Trusted Execution Technology

AppArmor: MAC を実装している Linux セキュリティモジュール

cgroups: リソース使用量を制御するための Linux カーネル機能

MAC ポリシー: 強制アクセス制御は SELinux または他のオペレーティングシステムを用いて実装されます

* この表にある機能はすべてのハイパーバイザーに適用できないかもしれません。また、ハイパーバイザー間で直接対応付けできないかもしれません。

第41章 仮想化層のセキュリティ強化

本章の初めに、インスタンスによる物理ハードウェアと仮想ハードウェアの両方の使用、関連するセキュリティリスク、それらのリスクを軽減するためのいくつかの推奨事項について議論します。SELinux 強制アクセス制御を仮想化コンポーネントと統合するためのオープンソースプロジェクトである sVirt の議論で本章を終わります。

物理ハードウェア（PCI パススルー）

多くのハイパーバイザーは PCI パススルーとして知られる機能を提供します。これにより、インスタンスがノードにあるハードウェアの一部に直接アクセスできます。たとえば、インスタンスがハイパフォーマンスコンピューティング用の compute unified device architecture（CUDA）を提供するビデオカードにアクセスするために使用されます。この機能は 2 種類のセキュリティリスクをもたらします。ダイレクトメモリアクセスとハードウェア感染です。

ダイレクトメモリアクセス（DMA）は、特定のハードウェアがホストコンピューターで任意の物理メモリアドレスにアクセスできる機能です。ビデオカードはときどきこの機能を有しています。しかしながら、インスタンスは指定された任意の物理メモリアクセスをすべきではありません。なぜなら、これはホストシステムと同じノードで実行している他のインスタンスを完全に表示できるかもしれないからです。ハードウェアベンダーはこれらの状況で DMA アクセスを管理するために input/output memory management unit（IOMMU）を使用します。そのため、クラウドアーキテクトは、ハイパーバイザーがこのハードウェア機能を使用するよう設定されていることを確実にすべきです。

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

注記

IOMMU 機能は、Intel により VT-d、AMD により AMD-Vi として提供されています。

ハードウェア感染は、インスタンスが悪意のある変更をファームウェアやデバイスの他の部分に行うときに発生します。このデバイスは他のインスタンス、またはホスト OS により使用されるため、悪意のあるコードはこれらのシステムの中に拡散する可能性があります。最終的な結果として、あるインスタンスがセキュリティドメインの範囲外で実行できます。これは何らかのハードウェアを共有している状況における潜在的な問題です。仮想ハードウェアよりも物理ハードウェアの状態をリセットすることが難しいため、この問題はこの状況に特有のものです。

ハードウェア感染問題の解決策はドメイン固有です。戦略はインスタンスがどのようにしてハードウェア状態を修正可能かを特定する事、その後インスタンスがハードウェアを使用している際に修正をリセットする方法を検知する事です。例えば、使用後のファームウェアの再度フラッシュが挙げられます。明らかに、いくつかのファームウェアは多数の書き込み後に故障するので、上記の作業はハードウェア寿命とセキュリティを天秤にかける必要があります。TPM 技術（link:Management/Node Bootstrappingで説明）は未承認のファームウェア変更を検知する解決策を提供します。選択した戦略に関わらず、この種のハードウェア共有に関するリスクを理解する事は、与えられたデプロイシナリオ用に適切にリスクを軽減する上で重要です。

加えて、PCI パススルーに関連したリスクと複雑性のため、これはデフォルトで無効化されるべきです。特定の用途のために有効化する場合、ハードウェアが再発行される前に確実にクリアするために、適切なプロセスを実行する必要があります。

仮想ハードウェア（QEMU）

仮想マシンの実行時、仮想ハードウェアは仮想マシンにハードウェアインターフェースを提供するソフトウェア層です。インスタンスは必要となるネットワーク、ストレージ、ビデオ、他のデバイスを提供するためにこの機能を使用します。これで覚えておくことは、お使いのほとんどのインスタンスは排他的に仮想ハードウェアを使用することです。一部はハードウェアに直接アクセスする必要があります。主要なオープンソースのハイパーバイザーはこの機能のために QEMU を使用します。QEMU は仮想化プラットフォームのニーズを満たしますが、作成と維持することが非常に挑戦的なソフトウェアプロジェクトであるとわかってきました。QEMU の機能のほとんどは、多くの開発者が理解しにくい低レベルなコードで実装されています。さらに、QEMU により仮想化されるハードウェアには、独自の癖を持つレガシーデバイスが数多くあります。これを一括りにするので、QEMU はハイパーバイザー突破攻撃を含む多くのセキュリティ問題の元になってきました。

上記の理由として、QEMU 堅牢化の率先したステップの実行が重要である事が挙げられます。我々は３つの特定のステップを推奨しています。コードベースの最小化、コンパイラーの堅牢化、sVirt・SELinux・AppArmor 等の強制アクセス制御の使用です。

QEMU コードベースの最小化

古くからある 1 つのセキュリティ原則は、システムから未使用のコンポーネントを削除することです。QEMU はさまざまな種類の仮想ハードウェアデバイスをサポートします。しかしながら、少しのデバイスだけが指定されたインスタンスに必要になります。多くのインスタンスは virtio デバイスを使用します。しかし、いくつかのレガシーなインスタンスは、glance メタデータを使用して指定できる、特定のハードウェアにアクセスする必要があります。

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

クラウドアーキテクトは、どのデバイスがクラウドユーザーに利用可能であるかを判断すべきです。必要ないすべてのデバイスは QEMU から削除すべきです。この手順は、QEMU 設定スクリプトに渡されるオプションを変更した後で、QEMU を再コンパイルする必要があります。最新の完全なオプション一覧は、QEMU ソースディレクトリの中で ./configure --help を単に実行します。お使いの環境に必要なものを判断し、残りのオプションを無効化します。

コンパイラーのセキュリティ強化機能

次の手順は、コンパイラーのセキュリティ強化オプションを使用して QEMU をセキュリティ強化することです。最近のコンパイラーは、出力バイナリのセキュリティを改善するために、さまざまなコンパイル時オプションを提供します。これらの機能には、より詳細を以下で説明しますが、relocation read-only (RELRO)、Stack Canaries、never execute (NX)、position independent executable (PIE)、address space layout randomization (ASLR) があります。

ほとんどの最近の Linux ディストリビューションは、すでにコンパイラーのセキュリティ強化を有効化して QEMU をビルドしています。そのため、以下の情報を続ける前に、既存のバイナリを確認したいでしょう。この確認を手助けできるツールの 1 つは checksec.sh と呼ばれます。

- RELocation Read-Only (RELRO): 実行ファイルのデータ部分をセキュリティ強化します。全体 RELRO モードと部分 RELRO モードが gcc によりサポートされます。QEMU 完全 RELRO が最善の選択肢です。これにより、グローバルオフセットテーブルが読み込み専用になり、出力実行ファイルのプログラムデータセクションの前にさまざまな内部データ部分が置かれます。

- Stack Canaries: バッファーオーバーフロー攻撃を防ぐ役に立てるために、スタックに値を置き、それらの存在を検証します。

- Never eXecute (NX): Data Execution Prevention (DEP) としても知られています。実行ファイルのデータ部分を必ず実行できなくします。

- Position Independent Executable (PIE): 位置に依存しない実行ファイルを生成します。ASLR のために必要です。

- Address Space Layout Randomization (ASLR): コード領域とデータ領域の両方を確実にランダムにします。実行ファイルが PIE を用いてビルドされるとき、カーネルにより有効化されます (最近の Linux カーネルはすべて ASLR をサポートします)。

すべてを一緒に利用し、いくつか追加の有用な保護を追加して、QEMU コンパイル時に以下の gcc コンパイラーオプションを推奨します。

```
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"
```

コンパイラーが確実に適切なセキュリティ強化を動作させるようコンパイルした後で、お使いの QEMU 実行ファイルをテストすることを推奨します。

ほとんどのクラウド環境は QEMU のようなソフトウェアを手動でビルドしたくないでしょう。プロセスが確実に繰り返し可能であり、最終結果を簡単にクラウドにデプロイできるようにするために、パッケージを使用するほうが良いでしょう。以下の参考情報は、既存のパッケージにコンパイラーのセキュリティ強化オプションを適用することの詳細を提供します。

- DEB パッケージ: Hardening Walkthrough

- RPM パッケージ: How to create an RPM package

強制アクセス制御

コンパイラーのセキュリティ強化機能により、QEMU プロセスへの攻撃をより難しくできます。しかし、攻撃者が成功すると、攻撃の影響範囲を抑えたいでしょう。強制アクセス制御は、QEMU プロセスの権限を必要な範囲に制限することにより、これを実現します。これは sVirt / SELinux または AppArmor により実現できます。sVirt 利用時、SELinux はすべての QEMU プロセスが別々のセキュリティドメインで動作するよう設定されます。AppArmor は同様の機能を提供するよう設定できます。以下のインスタンス分離のセクションで sVirt の詳細を示します。

sVirt: SELinux + 仮想化

KVM は複数のテナントを分離する基本技術を提供します。カーネルレベルの独特のアーキテクチャーを用いて、National Security Agency (NSA) により開発されたセキュリティ機構です。開発の起源は 2002 年までさかのぼり、Secure Virtualization (sVirt) 技術は最近の仮想化向けの SELinux の応用技術です。SELinux は、ラベルに基づいた分離制御を適用するために設計され、仮想マシンのプロセス、デバイス、データファイル、それらの上で動作するシステムプロセス間の分離を提供するために拡張されました。

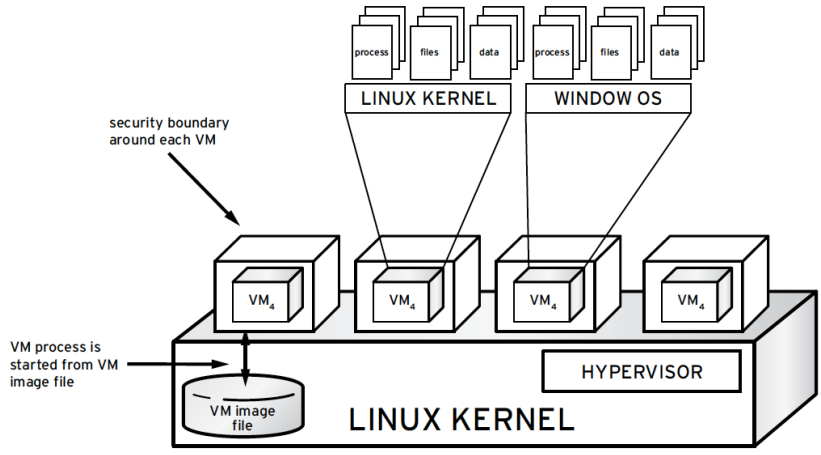
OpenStack の sVirt 実装は、2 種類の主要な脅威ベクターに対して、ハイパーバイザーホストと仮想マシンを保護することを目指しています。

- ハイパーバイザーの脅威 仮想マシンの中で動作している侵入されたアプリケーションは、バックエンドのリソースにアクセスするためにハイパーバイザーを攻撃します。たとえば、ホスト OS、アプリケーション、物理マシンにあるデバイスです。バックエンドの物理マシンが単一の仮想アプリケーションにある脆弱性のために侵入されうるため、これは仮想化に特有の脅威ベクターであり、考慮すべきリスクを表します。

- 仮想マシン（マルチテナント）の脅威 仮想マシンの中で動作している侵入されたアプリケーションは、他の仮想マシンとそのリソースにアクセスし、制御するためにハイパーバイザーを攻撃します。仮想マシンのイメージファイルの集合が単一のアプリケーションにある脆弱性のために侵入されうるため、これは仮想化に特有の脅威ベクターであり、考慮すべきリスクを表します。実ネットワークを保護するための管理技術が仮想マシン環境にそのまま適用できないため、この仮想ネットワークへの攻撃はおもな関心事です。

各 KVM ベースの仮想マシンは SELinux によりラベル付けされているプロセスです。これは各仮想マシンのセキュリティ境界を効率的に確立し

ます。このセキュリティ境界は、Linux カーネルにより監視され、強制されます。ホストマシンのデータファイルや他の仮想マシンのような、仮想マシンの境界外のリソースへのアクセスは制限されます。



上に示したとおり、sVirt による分離は仮想マシン内で動作しているゲストオペレーティングシステムに関わらず提供されます。Linux や Windows の仮想マシンを使用できます。さらに、多くの Linux ディストリビューションはオペレーティングシステム内の SELinux を提供しています。仮想マシンが内部の仮想リソースを脅威から保護できます。

ラベルとカテゴリ

KVM ベースの仮想マシンインスタンスは、svirt_image_t として知られる、独自の SELinux データタイプでラベル付けされています。カーネルレベルの保護により、悪意のあるソフトウェアのような権限のないシステムプロセスが、ディスクにある仮想マシンのイメージファイルを操作することを防ぎます。仮想マシンが電源オフのとき、イメージは以下のように svirt_image_t として保存されます。

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

svirt_image_t ラベルは独自にディスク上のイメージファイルを識別し、SELinux ポリシーがアクセス制限できるようにします。KVM ベースの Compute イメージが電源投入された際、sVirt はイメージに乱数の IDを付与します。sVirt は技術的には 1 ハイパーバイザーノードあたり

524,288 個の仮想マシンに数字IDを付与する事ができますが、OpenStack デプロイでこの制限に遭遇する事はまずないでしょう。

この例は sVirt カテゴリー識別子を示します。

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

ブーリアン

SELinux の管理負担を減らすために、多くのエンタープライズ Linux プラットフォームは sVirt のセキュリティ設定を簡単に変更するために、SELinux ブーリアンを利用します。

Red Hat Enterprise Linux ベースの KVM 環境は以下の sVirt ブーリアンを利用します。

sVirt SELinux ブーリアン	説明
virt_use_common	仮想化がシリアル通信ポートとパラレル通信ポートを使用することを許可します。
virt_use_fusefs	仮想化が FUSE マウントされたファイルを読み取ることを許可します。
virt_use_nfs	仮想化が NFS マウントされたファイルを管理することを許可します。
virt_use_samba	仮想化が CIFS マウントされたファイルを管理することを許可します。
virt_use_sanlock	制限された仮想マシンが sanlock を操作することを許可します。
virt_use_sysfs	仮想マシンがデバイス設定 (PCI) を管理することを許可します。
virt_use_usb	仮想化が USB デバイスを使用することを許可します。
virt_use_xserver	仮想マシンが X Window System と通信することを許可します。

第42章 ケーススタディ：インスタンス分離

このケーススタディでは、アリスとボブが所有するインスタンスが正しく分離されていることを確認する方法について説明します。まずはハイパーバイザーの選択とQEMUの強化、強制アクセスコントロールの適用について検討します。

アリスのプライベートクラウド

アリスは豊富な知識を持っている上、細かいポリシー強制のためにXen security module(XSM)を採用したいため、Xenをハイパーバイザーに選択します。

アリスはソフトウェアパッケージングとメンテナンスにそれなりのリソースを割り当てる予定です。これらのリソースを活用し、QEMUから多数コンポーネントを取り除くなど大幅カスタマイズをします。コンポーネントを取り除くことで、攻撃可能な部分は削減されます。また、QEMUのコンパイラ強化オプションがすべて有効になっていることも確認します。長期メンテナンスコストが増えてしまうことを理解した上でこれらの作業や設定を選択しています。

インスタンス間の分離を強めるため、アリスはXSMポリシー(Xen向け)とSELinuxポリシー(Linux domain0とデバイスドメイン向け)を作成しています。また、TPMのハイパーバイザーの起動を計測するためにXenに含まれるIntel TXTサポートを採用しています。

ボブのパブリッククラウド

パブリッククラウドではクレジットカードを所持していればユーザーになれるため、本質的に信頼性が低いことからボブはインスタンス分離に非常に気を遣っています。ボブはクラウドを配備するチームのメンバーの採用を開始したばかりなので、メンバー候補の検索条件として任意の特定スキルを指定できます。それを踏まえ、ボブは技術的な特徴、所持する証明等、コミュニティサポートを基準にハイパーバイザーを選択します。KVMはEAL 4+共通項目評価とラベル化されたセキュリティ保護プロフィール(LSPP)によってインスタンス分離を保証しています。これらの

強みとOpenStackコミュニティのKVMへの豊富なサポートを総合して考えた結果、ボブはKVMを採用することにします。

ボブはQEMUを再パッケージするために発生するコストを検討し、そのためのリソースをプロジェクトに割くことはできないと判断します。幸い、彼の使用しているLinuxのディストリビューションではコンパイラ強化オプションが有効になっているため、そのQEMUパッケージを採用します。最後に、仮想スタックに関わるSELinuxポリシーを管理するためにsVirtを利用します。

第43章 インスタンスのセキュリティサービス

仮想環境でインスタンスを運用する長所の一つは、ベアメタルで配備した際には利用できないセキュリティ管理方法の選択肢が増えることです。仮想スタック上のクラウドテナントの情報管理を改善する技術は多数存在します。

高いセキュリティ要件を持つOpenStackユーザーや配備者はこれらの技術の採用を検討すると良いかもしれませんが、状況によっては適用できない場合があります。クラウド運用においては、規範的なビジネス要件のために技術の選択肢が削られることがあります。また、run stateなど、仕組みによってはインスタンス内のデータを調べる機構もあり、システムのユーザーからは好まれないものもあります。

本章では、これらの仕組みの詳細とどのような状況においてインスタンスのセキュリティが向上されるかを説明します。また、プライバシー観点における懸念箇所にも焦点をあてます。データのパススルー、イントロスペクション、またエントロピー元の提供などが該当します。本セクションでは、下記のセキュリティサービスに焦点を当てます:

- インスタンスへのエントロピー
- ノードへのインスタンスのスケジューリング
- 信頼されたイメージ
- インスタンスのマイグレーション

インスタンスへのエントロピー

エントロピーとは、インスタンスがアクセスできるランダムデータの質と提供元のことを捉えています。暗号化技術は一般的にランダム性を採用しており、高品質なエントロピーのプールが必要です。通常、仮想マシンは十分なエントロピーを確保することが容易ではありません。エントロピー不足は一見まったく関係のないところで露見することがあります。例えば、インスタンスがSSHキーの生成を待っているため、ブートが遅くなることがあります。また、エントロピー不足を解決するために

ユーザーがインスタンス内部から低品質なエントロピー元を採用し、結果的にクラウド内で稼働するアプリケーションのセキュリティを下げることもあります。

これらの課題は、クラウドアーキテクトが高品質のエントロピーをクラウドインスタンスに提供することで対応できます。例えば、クラウド内にインスタンス用に適量なハードウェア乱数生成器(HRNG)があれば解決できます（適量はドメインによって異なる）。一般的なハードウェア乱数生成器なら通常運用されている50-100台のコンピュートノード分のエントロピーを生成することが可能です。高帯域ハードウェア乱数生成器（Intel Ivy Bridgeや最新プロセッサなどと提供されるRdRand instructionなど）はさらに多くのノードに対応できます。エントロピーの量が十分かどうかを判断するためには、クラウド上で運用するアプリケーションの要求を理解している必要があります。

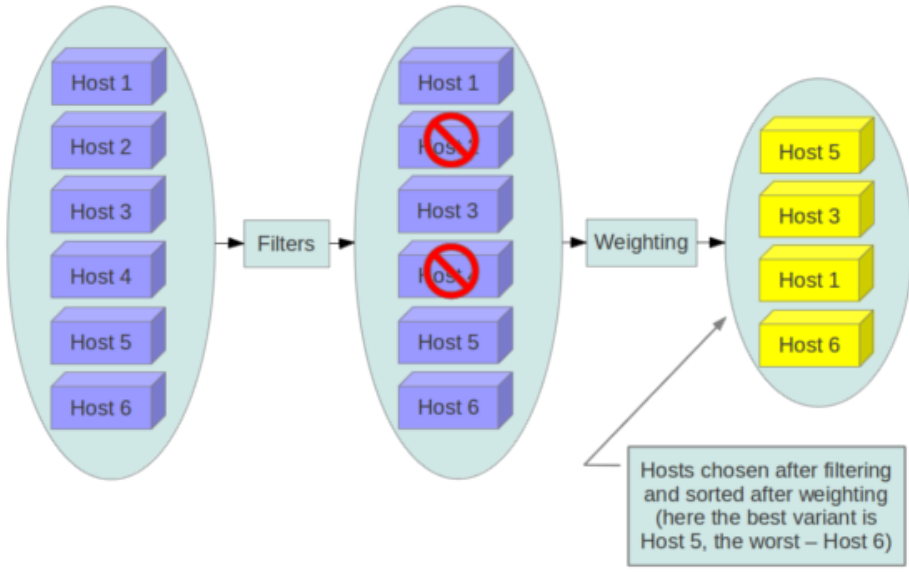
クラウド上でエントロピーが利用可能となったら、次はインスタンスからエントロピーを供給できるようにします。エントロピー収集デーモン(Entropy Gathering Daemon EGD)では、分散システム上でエントロピーを平等かつ安全な配布を実現しています。libvirtのエントロピー元としてEGDを使用するためのサポートも提供されています。

これらの機能に対して、コンピュートは未対応です。これらの機能との連携のため、開発者による実装の作業はあまり多く発生しないと思われます。

ノードへのインスタンスのスケジューリング

インスタンスを生成する前に、イメージのインスタンス化のためのホストを選択する必要があります。この選択はnova-schedulerによって行われ、さらにコンピュートとボリューム要求の伝達方法も決定します。

Grizzlyのデフォルトのnova schedulerはフィルタースケジュラーです。他にもスケジューラーは存在します(詳細はOpenStack Configuration ReferenceのScheduling を参照)。フィルタースケジューラーはフィルターと連携し、インスタンスの起動場所を決めます。このホスト選択作業があることによって、管理者は様々なセキュリティ要件を満たすことができます。クラウド配備種別によっては、次のような構成が組めます。例えばデータ分離が大きな懸念事項の場合、テナントのインスタンスは必ず同一のホスト上に配置するように設定できます。または、耐障害性のためにテナントのインスタンスをできるだけ異なるホスト上に配置するように設定できます。下の図は、フィルタースケジュラーの働きを表しています。



スケジューラーを提供しているすべてのOpenStackプロジェクトにおいて、スケジュールフィルタを使用することによってお客様やデータを分離できます。さらに安全ではないと判断されたクラウド上のマシンの破棄も行えます。クラウドを構築する際には、あらゆるセキュリティ目的のためにスケジュールフィルターの実装を選択できます。

下記ではセキュリティコンテキストにおいて役に立つ幾つかのフィルターを紹介します。OpenStack Configuration ReferenceのFilter Schedulerセクションにすべてのフィルター関連ドキュメントが掲載されています。要件に合わせてご参照ください。

テナントによるホスト全体予約

現在、ホスト全体予約のブループリントが公開されています。これによって効率面の負担はありますが、テナントは抱えるインスタンスのみのためにホストを確保できます。

ホストアグリゲート

ホストアグリゲート自体はフィルターではありませんが、管理者にマシンの集合体へキーバリューペアの割り当てを可能にします。これによってユーザーではなく、クラウド管理者によるコンピュートホストリソースの分配ができます。各ノードは複数のアグリゲートを持つことができ

ます。(詳細はOpenStack Configuration ReferenceのHost Aggregatesセクションを参照ください。)

AggregateMultiTenancyIsolation

テナントを特定のホストアグリゲート(集合体)に分離します。ホストがfilter_tenant_idというメタデータキーを持つアグリゲートの場合、そのテナント(あるいはそのテナント一覧)のみからインスタンスを作成します。ホストは複数のアグリゲートに所属することができます。メタデータキーを持たないアグリゲートに属する場合、すべてのテナントからインスタンスを作成できます。

DifferentHostFilter

特定のインスタンスのグループとは異なるホスト上にインスタンスをスケジュールします。このフィルタを利用するには、要求時にスケジュール情報としてキーにdifferent_hostを指定し、値にはインスタンスuuidのリストを渡す必要があります。SameHostFilterとは反対の働きを持つフィルターです。

GroupAntiAffinityFilter

GroupAntiAffinityFilterはグループに含まれるすべてのインスタンスは異なるホストで稼働していることを保証します。このフィルタを利用するには、要求時にスケジュール情報としてキーにgroupを指定し、値にはインスタンスuuidのリストを渡す必要があります。

信頼済コンピュートプール

Intel TXTを使用したシステムから送られたPCRの認証によってスケジューラーの対応を定義するためにOpen Attestation Project (OATS)と連携するスケジューラーフィルターがあります。

OpenAttestationエージェントはベンダー依存ではないTrouSerSライブラリを採用していますが、本機能は類似したAMD社のSEMと互換性があるかは不明です。

信頼されたイメージ

ユーザーはインストール済イメージあるいは自身がアップロードしたイメージを使用します。どちらの場合においても、採用したイメージは改ざんされていないことを確認したいでしょう。確認のためには、正式版のチェックサムなどの検証用情報と稼働しているイメージの証明情報が

必要です。このセクションでは、イメージの扱いに関するベストプラクティスと関連する既知の課題について説明します。

イメージ作成プロセス

OpenStackが提供するドキュメントではイメージの作成とGlanceへのアップロード方法について説明しています。ただし、オペレーティングシステムのインストールや強化のための設定方法やプロセスに関しては利用者が既に知識を持っていると想定しています。参考として、アップロード前にイメージがセキュアかどうかを確認するため情報を下記に説明します。また、イメージの採取には様々な方法があり、それぞれにおいてイメージの出所を検証するための独自の手順があります。

最初の選択肢は、信頼された提供元からブートメディアを入手することです。

```
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
```

次の選択肢は、OpenStack Virtual Machine Image Guideの活用です。こちらの場合、あなたの所属する組織のOS強化ガイドラインやRHEL6 STIGのような信頼性の高い第三者団体が提供するガイドラインに従うことを推奨します。

最後の手段として説明するのはイメージの自動生成機構の使用です。次の例では、Oz image builderを採用しています。OpenStackコミュニティでは、disk-image-builderというさらに新しいツールが公開されています。本ツールはセキュリティ観点において未検証です。

OzでNIST 800-53 セクションAC-19(d) の実装を手助けするRHEL 6 CCE-26976-1の例

```
<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>
```

本ガイドでは、手動イメージ構築プロセスは複雑で人為ミスが生まれやすいため推奨していません。OzやDisk-image-builderのような自動システム、ブート後のイメージ強化のためにChefやPuppetのような構成管理ツールなどを採用することによって一貫したイメージの作成だけでなく、時間が経過しても強化ガイドラインとベースイメージのコンプライアンス追跡が可能です。

パブリッククラウドサービスを使用する場合、クラウドプロバイダーにデフォルトイメージの作成プロセスのアウトラインを確認することを推奨します。また、自身が作成したイメージのアップロードが可能な場合、起動する前にイメージに変更が加えられていないかを確認したいでしょう。これらの手順については、イメージの出所に関する次のセクションを参照してください。

イメージの出所と妥当性確認

残念ながら、現在はインスタンス起動直前にコンピュートにイメージのハッシュを検証を強制する方法がありません。状況を理解するために、

イメージ起動の際にイメージがどのように扱われるのかを簡単に説明します。

イメージはGlanceサービスからノードのNovaサービスへ供給されます。この転送はSSLによって保護されている必要があります。イメージがノードに転送されたら、一般的なchecksumで検証され、起動するインスタンスのサイズに合わせてディスクが拡張します。以降、このノードで同じサイズの同一イメージを起動する場合はこの拡張されたイメージから起動されます。拡張されたイメージは起動前に再検証されないため、改ざんの可能性があります。これでは作成されたイメージのファイルの手動確認以外に確認方法がありません。

将来的にコンピュートまたはイメージサービスでインスタンス起動の前にイメージのハッシュを検証する機構を提供することが期待されています。さらに考えられる強力な代替手段はユーザーにイメージを署名させ、インスタンスの起動前に署名の検証を実行させることです。

インスタンスのマイグレーション

OpenStackと下層の仮想レイヤーによってOpenStackノード間のイメージのライブマイグレーションを実現しています。これにより、インスタンスのダウンタイムなくOpenStackコンピュートノードのシームレスなローリングアップデートが可能です。ただし、ライブマイグレーションにはそれなりのリスクが伴うことを注意する必要があります。リスクを理解するために、ライブマイグレーションの動作を理解することが重要です。次はライブマイグレーションの際のおおまかな流れを紹介しています。

1. 目的先ホストでインスタンスを起動

2. メモリを転送

3. ゲスト&syncディスクを停止

4. 転送状態となる

5. ゲストを起動

ライブマイグレーションのリスク

ライブマイグレーションのステージによっては、インスタンスのランタイムメモリやディスクの今テンスが平文でネットワーク上転送されます。そのため、ライブマイグレーション中には対処が必要なリスクがあります。次は一部のリスクの詳細を列挙しています。

- Denial of Service (DoS) : マイグレーションプロセス中に何かが失敗した場合、インスタンスを失う可能性があります。
- データの公開 : メモリやディスクの転送は安全に行う必要があります。
- データの操作 : メモリやディスクの転送が安全ではない場合、マイグレーション中にユーザーデータが攻撃者によって改ざんされる可能性があります。
- コードの挿入 : メモリやディスクの転送が安全ではない場合、マイグレーション中に攻撃者によってディスクやメモリ上の実行ファイルが操作される可能性があります。

ライブマイグレーションのリスクの軽減

ライブマイグレーションに関連するリスクを軽減するためには様々な手法があります。次のリストで詳しく説明します。

- ライブマイグレーションの無効化
- マイグレーションネットワークの分離
- ライブマイグレーションの暗号化

ライブマイグレーションの無効化

現在、OpenStackではデフォルトでライブマイグレーションを有効にしています。ライブマイグレーションはnova policy.jsonファイルへ下記の行を追加することによって無効化できます。

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

マイグレーションネットワーク

一般的には、ライブマイグレーションで発生するトラフィックは管理セキュリティドメインに制限するべきです。平文であり、稼働中のインスタンスのディスクとメモリを転送するということを踏まえると、安全性を確保するためにはライブマイグレーションのトラフィックを専用のネットワークに分離することを推奨します。専用ネットワークにトラフィックを分離することで、露出の危険性を下げることができます。

ライブマイグレーションの暗号化

あなたのユースケースでライブマイグレーションが有効な場合、libvirtdによるトンネル化、暗号化されたライブマイグレーションが行えます。ただし、この機能は現在のOpenStackダッシュボードやnova-clientコマンドで実装されておらず、libvirtdの手動設定のみでしか利用できません。暗号化されたライブマイグレーションと通常のライブマイグレーションの違いは、次のとおりです。最初に、稼働しているハイパーバイザーからインスタンスのデータをlibvirtdへコピーします。次に、両ホストのlibvirtdプロセス間に暗号化されたトンネルが作成されます。最後に、目的先libvirtdプロセスがインスタンスを下層のハイパーバイザーへコピーします。

第44章 ケーススタディ：インスタンス管理

このケーススタディでは、アリスとボブがインスタンスのエントロピー、インスタンスのスケジューリング、信頼できるイメージ、インスタンスのマイグレーションを尊重しつつ、彼らのクラウドを設計する方法について議論します。

アリスのプライベートクラウド

アリスはインスタンス群に高い品質の多くのエントロピーに対するニーズがあります。このため、彼女は各 compute ノード上で RdRand 命令をサポートする Intel Ivy Bridge チップセットを持つハードウェアの購入を決めました。エントロピー収集デーモン（EGD）と LibVirt の EGD サポートを使用して、Alice はこのエントロピープールが各 compute ノード上のインスタンスに配信されるようにします。

インスタンススケジューリングでは、全てのクラウド負荷が適切な起動時間保証を示すノードにデプロイされるようにする為、アリスは信頼できる compute プールを使用します。クラウド中で使用されるイメージがクラウド管理者に既知で信頼できる方法で作成されたものである事を保証するため、アリスはユーザにイメージをアップロードする権限を与えない事を決めました。

最後に、アリスはインスタンスのマイグレーションを無効化しました。この機能はこのクラウドで実行される予定の高パフォーマンスアプリケーション負荷にはほとんど不要だからです。これにより、インスタンスマイグレーションにまつわる様々なセキュリティ関連を避ける事ができます。

ボブのパブリッククラウド

ボブは、金融業界の企業ユーザの幾つかにとってエントロピーが重要となる事を理解しています。しかしながら、費用と複雑さが増える為、ボブは彼のクラウドの初回導入分にハードウェアエントロピーの導入を見送る事を決めました。彼は自分の２世代目のクラウドアーキテクチャに向けた後の改善では、早期のフォローとしてハードウェアエントロピーを追加します。

ボブは、顧客が高品質なサービスを受けられるようにする事に興味があります。彼は、インスタンススケジューリングを超えた過剰なほど明確なユーザコントロールの提供が、サービス品質（QoS）にマイナス影響を与える事を心配しています。ですので、この機能を無効化しました。ボブは使用するユーザに対して既知の信頼できるソースからのクラウド中のイメージを提供します。加えて、彼はまた、ユーザに自分のイメージアップロードを許可します。しかしながら、ユーザは一般に自分のイメージを共有できません。これは、クラウド中の他のユーザのセキュリティにマイナスインパクトを与えかねない、悪意あるイメージを共有する事からユーザを守る助けになります。

マイグレーションでは、ボブは最小のユーザダウンタイムでのローリングアップデートをサポートする為に、安全なインスタンスマイグレーションを有効にしたいと思っています。ボブは、全てのマイグレーションが独立した VLAN 上で実行されるようにします。彼は、Nova クライアントツールが暗号化マイグレーションをより良くサポートするまで暗号化マイグレーションの実装を遅らせる計画を立てています。

第45章 フォレンジングとインシデント対応

多数の活動がクラウド環境内で行われます。これはハードウェア、オペレーティングシステム、仮想マシンマネージャ、OpenStackサービス群、インスタンス作成やストレージアタッチのようなクラウド⇔ユーザ活動、全体の土台であるネットワーク、最後に様々なインスタンス上で実行されるアプリケーションを使用するエンドユーザのミックスです。

ログの生成と収集は OpenStack インフラのセキュリティ監視の重要なコンポーネントです。ログは日々の管理者・テナント・ゲストの行動に加え、あなたの OpenStack デプロイを構成する Compute、Networking、ストレージ、他のコンポーネントの活動の可視性を提供します。

ロギングの基本: ログを集中収集する方法と同様、設定、ログレベル設定、ログファイルの位置、ログの使用とカスタマイズ方法は、OpenStack Operations Guide で充分にカバーされています。

ログは率先したセキュリティや継続的なコンプライアンス活動に有用であるのみならず、インシデントの調査と対応の為の情報源としても有用です。

例えば、Identity サービスまたはその代替認証システムへのアクセスログ解析は、アカウント等を選択してイベントを制限する／しないで、失敗したログイン、それらの頻度、アクセス元IPアドレスを警告します。ログ解析は検知をサポートします。

検知時、追加のアクションになるのは、IP のブラックリストだったり、ユーザのパスワードを補強する事を推奨したり、ユーザアカウントが休眠状態である場合はその無効化でさえあったりします。

監視ユースケース

イベントの監視はより率先的で、リアルタイムの検知と対応を提供します。監視の助けとなるいくつかのツールがあります。

OpenStack クラウドインスタンスの場合、ハードウェア、OpenStack サービス、クラウドリソース使用量を監視する必要があります。最後

は、柔軟性、ユーザの変化するニーズへのスケール性への要求から生じるものです。

ここで、ログ収集、解析、監視を実装する際に考慮すべき重要なユースケースがいくつかあります。これらのユースケースは、様々な商用やオープンソースのツール、自作のスクリプト等を通じて実装・監視できます。これらのツールとスクリプトは、電子メールや組み込まれたダッシュボードで管理者に送信されるイベントを生成できます。あなたの場合のネットワークに適用できる追加のユースケースや、変則的な挙動を考慮できるようにするものを考慮する事は重要です。

- ログ生成無しの検知は価値の高いイベントです。このようなイベントはサービス障害、または一時的にログをオフにしたり、監視者から隠れるためにログレベルを変更した侵入者を示している可能性があります。

- スケジュール外の start/stop のようなアプリケーションイベントは、潜在的なセキュリティ的なウラについての監視と確認作業を行うイベントでもあるでしょう。

- ユーザログイン、再起動のような OpenStack サービスマシン上の OS イベントもまた、使用／誤用への価値ある洞察を与えます。

- OpenStack サーバ群の負荷を検知可能にする事はまた、高可用化対応の為に負荷分散用追加サーバを導入する為の対応を可能にする事でもあります。

- 行動可能な他のイベントはネットワークブリッジがダウンした事です。compute ノード上で設定がクリアされた iptables や、インスタンスへのアクセスの重大なロスはユーザを不幸にします。

- identity サービス中のユーザ／テナント／ドメイン削除に伴う見捨てられたインスタンスかあのセキュリティリスクを低減する為、システム中で通知を生成する事と、インスタンス削除、アタッチしたボリュームの切断、CPU やストレージリソースの回収等のイベントに適切に対応する OpenStack コンポーネントを用意する事について議論があります。

クラウドには多数の仮想インスタンスがあり、これらのインスタンスの監視はハードウェア監視と CRUD イベントのみ含むログファイルの背後にあります。

侵入検知ソフトウェア、アンチウイルスソフトウェア、スパイウェア検知・削除ユーティリティのようなセキュリティ監視制御は、攻撃や侵入が発生した時と方法を示すログを生成できます。クラウドマシン上にこ

れらのツールをデプロイする事は、価値と保護を提供します。（クラウド上でインスタンスを実行する）クラウドユーザも自身のインスタンス上でこのようなツールを実行したいかも知れません。

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 ケーススタディ: 監視とログ採取

このケーススタディでは、アリスとボブがパブリッククラウドとプライベートクラウドの中で監視とロギングを実行する方法を議論します。どちらのインスタンスでも、時間同期とログの集中保存が適切なアセスメントの実施と変則的な事のトラブル対応に極めて重要となります。単なるログの収集はそれほど有用ではなく、利用できるイベントを生成する為のロバストな監視システムを構築する必要があります。

アリスのプライベートクラウド

プライベートクラウドでは、アリスはテナントの要件についてより深く理解しており、そのため監視やログ採取上で適切な全景や法令遵守を追加できます。アリスは重要なサービスとデータを認識し、少なくともこれらのサービス上でログが採取されるようにし、中央ログサーバにログが集約されるようにする必要があります。また、障害可能性の数を制限する為、相互関係・警告を実装する必要があります。相互関係と警告を実装するために、アリスはログデータを彼女の組織の既存の SIEM ツールに送信します。セキュリティ監視は継続プロセスであり、ネットワークトラフィック活動と使用量についてより深く理解する為に、アリスはユースケースと警告の定義を継続する必要があります。

ボブのパブリッククラウド

ログが作成された際、（パブリッククラウドプロバイダとして）法律順守と状況判断の両方で、ボブはログ採取に興味があります。これはつまり、ボブの顧客のコンプライアンス監査の為、彼らの代わりにタイムリーかつ関連のあるログ又はレポートを提供する為の彼の能力と同様、コンプライアンスはプロバイダとしてのボブが従うべきものであるという事です。それを念頭に置いて、ボブは彼のインスタンス、ノード、インフラデバイス全てで外部の良好と知られている時間デバイスを用いて時間同期を実行するよう設定しています。加えて、ボブのチームは彼の顧客用に、ボブの SIEM ツールからセルフサービスでログ取得を実行する為の Django ベースの Web アプリケーションを構築しています。ボブは、顧客とクラウド管理者の双方に運用判断を提供する為、ロバストな警告セットがあり、彼の CMDB インテグレーションを持つSIEM ツールも使用します。

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 監査プロセスの理解

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 - サービス提供組織がその管理について説明し、その公正さをレポートします。特定時点で関連する管理対象を統制できているか、その設計の持続可能性も報告します。

- Type 2 - サービス組織が統制対象を統制するために使用するシステム、設計の持続性、および運用効率性に関する管理者の説明内容が公正かをレポートします。特定期間を通しての説明も必要です。

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 - サービス提供組織がその管理について説明し、その公正さをレポートします。特定時点で関連する管理対象を統制できているか、その設計の持続可能性も報告します。

- Type 2 - サービス組織が統制対象を統制するために使用するシステム、設計の持続性、および運用効率性に関する管理者の説明内容が公正かをレポートします。特定期間を通しての説明も必要です。

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。

HIPPAは認証ではなく、カルテ情報の保護に関するガイドラインです。PCI-DSSと似ています。PCIとHIPAAの両方でもっとも重要な課題は、クレジットカード情報とカルテ情報が流出しないようにすることです。クラウドプロバイダーによる流出があった場合、PCIとHIPAAの統制下において検査されます。その内容が遵守に足るものであれば、そのプロバイダーはすみやかに是正措置の実行と情報流出の通知、およびコンプライアンス活動予算の大幅な追加を期待されます。もし足るものでなければ、現地での査察、罰金、merchant ID (PCI)の失効、および評判に大きな傷がつくことが予想されます。

カルテ情報を所有するユーザーや組織はHIPPAの要件をサポートし、HIPAA対象事業者となる必要があります。もしこの事業者がサービスを、この場合は対象のOpenStackクラウドがカルテ情報を利用、保存、アクセスしうるのであれば、HIPAA Business Associate Agreement - BAAの締結が必要です。BAAはHIPAA対象事業者と、HIPAA要件に従ってカルテ

情報を扱っているOpenStackサービスプロバイダーの間で締結されます。もしサービスプロバイダーがセキュリティ統制、強化を怠るなど、カルテ情報を要件通りに扱っていなければHIPAAの罰金や罰則が適用されることがあります。

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。
- 統制の適用 システムのセキュリティが特定されれば、OpenStackアーキテクトは選択した統制を適用するために、NIST 800-53を活用します。たとえば、安全なパスワードの構成を仕様化するなど。

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。

彼女のプライベートクラウドで使われる全ての技術は、NIST 800-53とFedRAMPに従い、FIPS認証技術であることが求められます。米国国防省が関わる場合、国防省のIA - Information AssuranceおよびIA-enabled対象機器/システムの構成標準であるSecurity Technical Implementation Guides (STIGs) も関係します。OpenStack向けのSTIGが無くとも、アリスはさまざまな要素を考慮し、各OpenStackサービス毎に、いくつかの潜在的な要件を考慮しなければいけません。たとえば、networking SRG - Security Requirements GuidesとApplication SRGはどちらも対象です(list of SRGs)。他の重要な統制として、クラウド内の全てのIDではPKIが使われ、SELinuxが有効であり、すべての全ての通信経路が暗号化で

き、持続的に監視が行われ、かつ明快に文書化されていること、などが挙げられます。なお、アリスはオブジェクトの暗号化を考慮しませんでしたが、これはプロバイダーというよりは、テナントの責任であるからです。

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
- Command-Line Interface Reference

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 (savanna)
- 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

アクセス制御リスト
: オブジェクトに対する権限の一覧。オブジェクトに対して、アクセスできるユーザーやシステムプロセスを特定する。また、特定のオブジェクトに対してどのような操作が行えるかを定義する。アクセス制御リスト（ACL）の一般的な項目では対象項目と操作を指定する。例えば、1つのファイルに対して(Alice, delete)というACL項目が定義されると、Aliceにファイルを削除する権限が与えられる。

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
: OpenStackがサポートするオペレーティングシステム構成管理ツール。

Qpid
: OpenStackでサポートされているメッセージキューのソフトウェア。RabbitMQの代替品。

RabbitMQ
: OpenStackでデフォルトで採用されているメッセージキューのソフトウェア。

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
: OpenStack によりサポートされるメッセージキューソフトウェア。RabbitMQ の代替。0MQ とも表記。